Manual do usuário

AWS Glue

AWS Glue: Manual do usuário

Table of Contents

O que é o AWS Glue?

O AWS Glue é um serviço de integração de dados com tecnologia sem servidor que facilita aos usuários de análise a descoberta, preparação, transferência e integração de dados de várias fontes. Você pode usá-lo para análise, machine learning e desenvolvimento de aplicações. Também inclui outras ferramentas de produtividade e operações de dados para criação, execução de trabalhos e implementação de fluxos de trabalho de negócios.

Com o AWS Glue, você pode detectar e se conectar a mais de 70 fontes de dados diversas e gerenciar seus dados em um catálogo de dados centralizado. Você pode criar, executar e monitorar visualmente pipelines de extração, transformação e carregamento (ETL) para carregar dados em seus data lakes. Além disso, é possível pesquisar e consultar imediatamente os dados catalogados usando o Amazon Athena, o Amazon EMR e o Amazon Redshift Spectrum.

O AWS Glue consolida os principais recursos de integração de dados em um único serviço. Isso inclui descoberta de dados, ETL moderno, limpeza, transformação e catalogação centralizada. Também conta com tecnologia sem servidor, o que significa que não há infraestrutura para gerenciar. Com suporte flexível para todas as workloads, como ETL, ELT e transmissão em um único serviço, o AWS Glue oferece suporte a usuários em várias workloads e tipos de usuários.

Além disso, o AWS Glue facilita a integração de dados em sua arquitetura. Ele se integra aos serviços de análise da AWS e a data lakes do Amazon S3. O AWS Glue tem interfaces de integração e ferramentas de criação de trabalhos que são descomplicadas para todos os usuários, de desenvolvedores a usuários corporativos, com soluções personalizadas para conjuntos variados de habilidades técnicas.

Com a capacidade de escalar sob demanda, AWS Glue ajuda você a se concentrar em atividades de alto valor que maximizam o valor de seus dados. Ele pode ser escalado para qualquer tamanho de dados e oferece suporte a todos os tipos de dados e variações de esquema. Para aumentar a agilidade e otimizar custos, o AWS Glue fornece alta disponibilidade integrada e cobrança com pagamento conforme o uso.

Para obter informações sobre preços, consulte AWS Glue preços.

AWS Glue Studio

O AWS Glue Studio é uma interface gráfica que facilita a criação, a execução e o monitoramento de trabalhos de integração de dados no AWS Glue. Você pode compor visualmente fluxos de trabalho

de transformação de dados e executá-los perfeitamente no mecanismo de ETL com tecnologia sem servidor baseado no Apache Spark do AWS Glue.

Com o AWS Glue Studio, você pode criar e gerenciar trabalhos que coletam, transformam e limpam dados. Use também o AWS Glue Studio para solucionar problemas e editar scripts de trabalho.

Tópicos

Recursos do AWS Glue

Os recursos do AWS Glue se enquadram em três categorias principais:

- Descobrir e organizar dados
- Transformar, preparar e limpar dados para análise
- Criar e monitorar pipelines de dados

Descobrir e organizar dados

- Unifique e pesquise em vários armazenamentos de dados: armazene, indexe e pesquise em várias fontes e coletores de dados catalogando todos os seus dados na AWS.
- Descubra dados automaticamente: use os crawlers do AWS Glue para inferir automaticamente as informações do esquema e integrá-las ao AWS Glue Data Catalog.
- Gerencie esquemas e permissões: valide e controle o acesso a bancos de dados e tabelas.
- Conecte-se a uma ampla variedade de fontes de dados: acesse várias fontes de dados, tanto on-premises como na AWS, usando conexões do AWS Glue para criar seu data lake.

Transformar, preparar e limpar dados para análise

- Transforme visualmente dados com uma interface de tela de trabalho: defina seu processo de ETL no editor de trabalhos visuais e gere automaticamente o código para extrair, transformar e carregar dados.
- Crie pipelines de ETL complexos com agendamento de tarefas simples: invoque trabalhos do AWS Glue em um cronograma, sob demanda ou com base em um evento.
- Limpe e transforme a transmissão de dados em trânsito: possibilite o consumo contínuo de dados, limpe-os e transforme-os em trânsito. Isso os disponibiliza para análise em segundos no datastore de destino.
- Elimine a duplicação e limpe dados com machine learning integrado: limpe e prepare dados para análise sem se tornar um especialista em machine learning usando o recurso `FindMatches`. Esse recurso elimina a duplicação e encontra registros que são correspondências imperfeitas entre si.
- Blocos de anotação de trabalho integrados: os blocos de anotação de trabalho do AWS Glue fornecem blocos de anotação com tecnologia sem servidor com configuração mínima no AWS Glue para que você comece a usar rapidamente.
- Edite, depure e teste o código ETL: com as sessões interativas do AWS Glue, você pode explorar e preparar dados de forma interativa. Você pode explorar, experimentar e processar dados de forma interativa usando o IDE ou o bloco de anotações de sua preferência.
- Defina, detecte e corrija dados sigilosos: a detecção de dados sigilosos AWS Glue permite definir, identificar e processar dados sigilosos no pipeline de dados e no data lake.

Criar e monitorar pipelines de dados

- Escale automaticamente de acordo com a workload: de maneira dinâmica, aumente e diminua verticalmente a escala dos recursos de acordo com a workload. Isso atribui operadores a trabalhos somente quando necessário.
- Automatize trabalhos com acionadores baseados em eventos: inicie os crawlers ou trabalhos do AWS Glue com acionadores baseados em eventos e crie uma cadeia de trabalhos e crawlers dependentes.
- Execute e monitore trabalhos: Execute trabalhos do AWS Glue com o mecanismo de sua escolha, Spark ou Ray. Monitore-os com ferramentas de monitoramento automatizadas, insights de execução de trabalhos do AWS Glue e o AWS CloudTrail. Aprimore o seu monitoramento de trabalhos apoiados pelo Spark com a interface do usuário do Apache Spark.
- Defina fluxos de trabalho para atividades de ETL e integração: defina fluxos de trabalho para ETL e atividades de integração para vários crawlers, trabalhos e acionadores.

Aprender sobre inovações no AWS Glue

Saiba mais sobre as inovações mais recentes no AWS Glue e ouça como os clientes usam o AWS Glue para permitir a preparação de dados de autoatendimento em toda a organização.

Saiba mais sobre como os clientes escalam o AWS Glue além da configuração tradicional e como eles configuram o AWS Glue para monitoramento de trabalho e a performance.

Conceitos básicos do AWS Glue

Recomendamos que você inicie por estas seções:

- Visão geral do uso do AWS Glue
- Conceitos do AWS Glue
- Configurar permissões do IAM para o AWS Glue
- Conceitos básicos do AWS Glue Data Catalog
- Criar trabalhos no AWS Glue
- Conceitos básicos das sessões interativas do AWS Glue
- Orquestração no AWS Glue

Como acessar o AWS Glue

Você pode criar, visualizar e gerenciar seus trabalhos do AWS Glue usando as seguintes interfaces:

- Console do AWS Glue: fornece uma interface da Web para você criar, visualizar e gerenciar trabalhos do AWS Glue. Para acessar o console, acesse AWS Glue.
- AWS Glue Studio: fornece uma interface gráfica para criar e editar seus trabalhos do AWS Glue visualmente. Para obter mais informações, consulte O que é o AWS Glue Studio.
- Seção sobre o AWS Glue na referência da AWS CLI: fornece comandos da AWS CLI que você pode usar com o AWS Glue. Para obter mais informações, consulte a referência da AWS CLI para o AWS Glue.
- API do AWS Glue: fornece uma referência de API completa para desenvolvedores. Para obter mais informações, consulte a API do AWS Glue.

Serviços relacionados

Os usuários do AWS Glue também utilizam:

- AWS Lake Formation : serviço que é uma camada de autorização que fornece controle de acesso detalhado a recursos do AWS Glue Data Catalog.
- AWS Glue DataBrew : uma ferramenta de preparação de dados visuais que pode ser usada para limpar e normalizar dados sem escrever nenhum código.

AWS Glue: como funciona

O AWS Glue usa outros produtos da AWS para orquestrar seus trabalhos de ETL (extração, transformação e carregamento) para criar data warehouses e data lakes e gerar transmissões de saída. O AWS Glue chama operações de API para transformar seus dados, criar logs de runtime, armazenar a lógica do trabalho e criar notificações para ajudar você a monitorar as execuções de trabalhos. O console do AWS Glue conecta esses serviços em um aplicativo gerenciado, para que você possa se concentrar na criação e no monitoramento do seu trabalho de ETL. O console executa operações de desenvolvimento de trabalhos e administrativas em seu nome. Você fornece credenciais e outras propriedades para que o AWS Glue acesse suas fontes de dados e grave nos destinos de dados.

O AWS Glue provisiona e gerencia os recursos necessários para executar sua workload. Não é necessário criar a infraestrutura para uma ferramenta de ETL porque o AWS Glue faz isso por você. Quando recursos são necessários, o AWS Glue usa uma instância do grupo de instâncias para executar sua workload e reduzir o tempo de inicialização.

Com o AWS Glue, você cria trabalhos usando definições de tabela no Data Catalog. Os trabalhos consistem em scripts com a lógica de programação que executa a transformação. Você usa gatilhos para iniciar trabalhos em uma programação ou como resultado de um evento especificado. Você determina onde seus dados de destino residirão e quais dados de origem que preencherão seu destino. Com sua entrada, o AWS Glue gera o código necessário para transformar seus dados de origem em dados de destino. Você também pode fornecer scripts no console ou na API do AWS Glue para processamento dos seus dados.

Fontes e destinos de dados

O AWS Glue para Sparks permite que você leia e grave dados de vários sistemas e bancos de dados, incluindo:

- Amazon S3
- Amazon DynamoDB
- Amazon Redshift
- Amazon Relational Database Service (Amazon RDS)
- Bancos de dados acessíveis a JDBC de terceiros
- MongoDB e Amazon DocumentDB (compatível com MongoDB)
- Outros conectores do Marketplace e plug-ins do Apache Spark

Streams de dados

O AWS Glue para Sparks pode transmitir dados dos seguintes sistemas:

- Amazon Kinesis Data Streams
- Apache Kafka

O AWS Glue está disponível em diversas regiões da AWS. Para obter mais informações, consulte Regiões e endpoints do AWS no Referência geral da Amazon Web Services.

Tópicos

Trabalhos de ETL com tecnologia sem servidor executados em isolamento

O AWS Glue executa trabalhos de ETL em um ambiente de tecnologia sem servidor com sua escolha de mecanismo, Spark ou Ray. O AWS Glue executa esses trabalhos em recursos virtuais que ele provisiona e gerencia na sua própria conta de serviço.

O AWS Glue é projetado para fazer o seguinte:

- Diferenciar dados de clientes.
- Proteger os dados do cliente em trânsito e em repouso.
- Acesse os dados dos clientes apenas quando necessário, em resposta às solicitações deles, usando credenciais temporárias com escopo ou com o consentimento dos clientes para funções do IAM na conta deles.

Durante o provisionamento de um trabalho de ETL, você fornece fontes de dados de entrada e destinos de dados de saída na sua nuvem virtual privada (VPC). Além disso, você fornece a função

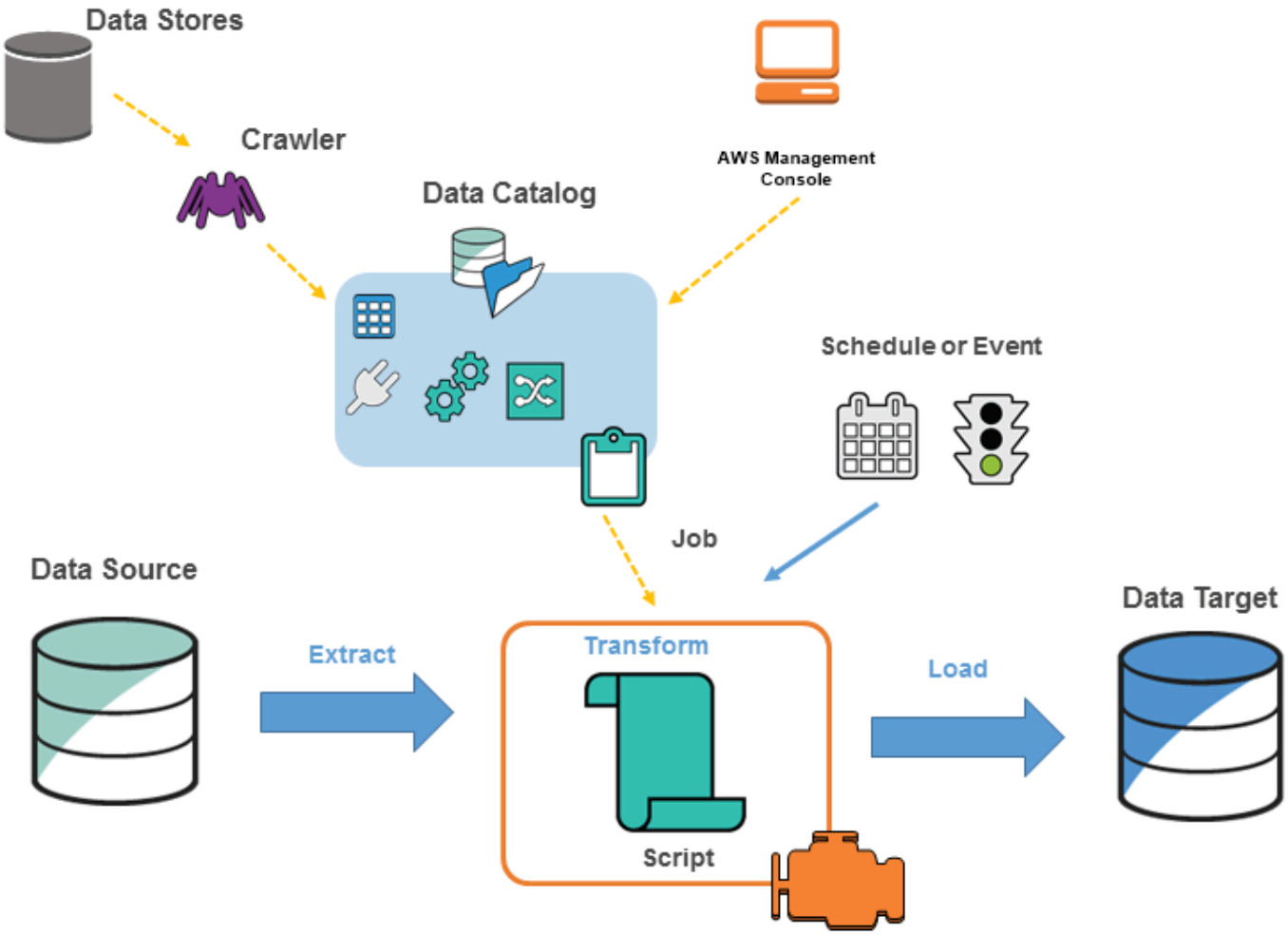
do IAM, o ID da VPC, o ID da sub-rede e o grupo de segurança necessários para acessar fontes e destinos de dados. Para cada tupla (ID da conta do cliente, perfil do IAM, ID da sub-rede e grupo de segurança), o AWS Glue cria um novo ambiente que é isolado na rede e um nível de gerenciamento de todos os outros ambientes dentro da conta de serviço do AWS Glue.

O AWS Glue cria interfaces de rede elástica na sua sub-rede usando endereços IP privados. Os trabalhos usam essas interfaces de rede elástica para acessar as fontes de dados e os destinos de dados. O tráfego de entrada e saída e dentro do ambiente de execução do trabalho é regido pelas políticas de VPC e de rede com uma exceção: as chamadas feitas para as bibliotecas do AWS Glue podem transmitir tráfego para as operações da API do AWS Glue por meio da VPC do AWS Glue. Todas as chamadas de API do AWS Glue são registradas. Dessa forma, os proprietários de dados podem auditar o acesso da API habilitando o AWS CloudTrail, que fornece logs de auditoria à sua conta.

Os ambientes gerenciados pelo AWS Glue que executam seus trabalhos de ETL são protegidos com as mesmas práticas de segurança seguidas por outros serviços da AWS. Para obter uma visão geral das práticas e responsabilidades de segurança compartilhadas, consulte o whitepaper Introduction to AWS Security Processes.

Conceitos do AWS Glue

O diagrama a seguir mostra a arquitetura de um ambiente do AWS Glue.



Você define os trabalhos no AWS Glue para atender aos requisitos de extração, transformação e carregamento (ETL) de dados de uma fonte de dados para um destino de dados. Você pode realizar as seguintes ações:

- Para fontes de armazenamento de dados, defina um crawler para preencher seu AWS Glue Data Catalog com definições da tabela de metadados. Você aponta seu crawler para um armazenamento de dados e ele cria definições de tabela no Data Catalog. Para fontes de transmissão, defina manualmente as tabelas do Data Catalog e especifique as propriedades de fluxo de dados.

  Além das definições de tabela, o AWS Glue Data Catalog contém outros metadados necessários para definir trabalhos de ETL. Usar esses metadados ao definir um trabalho de transformação dos seus dados.

- O AWS Glue pode gerar um script para transformar seus dados. Se preferir, você pode fornecer o script no console ou na API do AWS Glue.
- É possível executar trabalhos sob demanda ou configurá-los para iniciar quando um determinado gatilho for acionado. Os gatilhos podem ser programações ou eventos baseados em tempo.

  Quando seu trabalho é executado, um script extrai os dados da sua fonte de dados, transforma esses dados e os carrega no seu destino de dados. O script é executado em um ambiente do Apache Spark no AWS Glue.

> **Important**
>
> As tabelas e os bancos de dados contidos no AWS Glue são objetos no AWS Glue Data Catalog. Eles contêm metadados;, e não dados de um armazenamento físico.

Dados baseados em texto, como CSVs, devem ser codificados em **`UTF-8`** para que o AWS Glue processe-os com êxito. Para obter mais informações, consulte UTF-8 na Wikipédia.

Terminologia do AWS Glue

O AWS Glue depende da interação de vários componentes para criar e gerenciar o fluxo de trabalho de extração, transformação e carregamento (ETL).

AWS Glue Data Catalog

O armazenamento persistente de metadados no AWS Glue. Ele contém definições de tabela, definições de trabalho e outras informações de controle para o gerenciamento do ambiente do AWS Glue. Cada conta da AWS tem um AWS Glue Data Catalog por região.

Classificador

Determina o esquema dos seus dados. O AWS Glue fornece classificadores para tipos de arquivo comuns, como CSV, JSON, XML, AVRO e outros. Ele também fornece classificadores para sistemas comuns de gerenciamento do banco de dados relacional usando uma conexão JDBC. Você pode escrever seu próprio classificador usando um padrão grok ou especificando uma linha de tag em um documento XML.

Conexão

Um objeto do Data Catalog que contém as propriedades necessárias para se conectar a um armazenamento de dados específico.

Crawler

Um programa que se conecta a um armazenamento de dados (origem ou destino), passa por uma lista prioritária de classificadores para determinar o esquema dos dados e cria tabelas de metadados no AWS Glue Data Catalog.

Banco de dados

Um conjunto de definições da tabela associada do Data Catalog organizadas em um grupo lógico.

Datastore, fonte de dados, destino de dados

Um datastore é um repositório para armazenar seus dados persistentemente. Os exemplos incluem buckets do Amazon S3 e bancos de dados relacionais. Uma fonte de dados é um datastore que é usado como entrada para um processo ou transformação. Um destino de dados é um datastore no qual um processo ou transformação grava.

Endpoint de desenvolvimento

Um ambiente que pode ser usado para desenvolver e testar seus scripts de ETL do AWS Glue.

Quadro dinâmico

Uma tabela distribuída que oferece suporte a dados aninhados, como estruturas e matrizes. Cada registro é autodescritivo, projetado para flexibilidade de esquema com dados semiestruturados. Cada registro contém os dados e o esquema que descreve esses dados. Você pode usar quadros dinâmicos e DataFrames do Apache Spark em seus scripts de ETL, bem como converter entre eles. Os quadros dinâmicos fornecem um conjunto de transformações avançadas para limpeza de dados e ETL.

Trabalho

A lógica de negócios que é necessária para executar o trabalho de ETL. Ela é composta por um script de transformação, fonte de dados e destinos de dados. As execuções de trabalho são iniciadas por gatilhos que podem ser programados ou acionados por eventos.

Painel de performance do trabalho

O AWS Glue fornece um painel de execução abrangente para seus trabalhos de ETL. O painel exibe informações sobre execuções de trabalhos a partir de um período de tempo específico.

Interface do bloco de anotações

Uma experiência aprimorada de bloco de anotações com configuração com um clique para facilitar a criação de trabalhos e a exploração de dados. O bloco de anotações e as conexões são configurados automaticamente para você. Você pode usar a interface de caderno baseada na Juypter Notebook para desenvolver, depurar e implantar interativamente scripts e fluxos de trabalho usando a infraestrutura Apache Spark de ETL com tecnologia sem servidor do AWS Glue. Você pode realizar também consultas ad-hoc, análise de dados e visualização (por exemplo, tabelas e gráficos) no ambiente do bloco de anotações.

Script

Código que extrai dados de origens, transforma esses dados e os carrega em destinos. O AWS Glue gera scripts PySpark ou Scala.

Tabela

A definição de metadados que representa seus dados. Não importa se os seus dados estão em um arquivo do Amazon Simple Storage Service (Amazon S3), uma tabela do Amazon Relational Database Service (Amazon RDS) ou em outro conjunto de dados, uma tabela definirá o esquema dos seus dados. Uma tabela no AWS Glue Data Catalog consiste em nomes de colunas, definições de tipos de dados, informações de partição e outros metadados relacionados a um conjunto de dados de base. O esquema de dados é representado na sua definição da tabela do AWS Glue. Os dados reais permanecem no seu armazenamento de dados original, em um arquivo ou uma tabela de banco de dados relacional. O AWS Glue cataloga seus arquivos e tabelas de banco de dados relacional no AWS Glue Data Catalog. Eles são usados como fontes e destinos quando você cria um trabalho de ETL.

Transformação

A lógica de código que é usada para manipular seus dados em um formato diferente.

Trigger

Inicia um trabalho de ETL. Os gatilhos podem ser definidos com base em um horário/evento programado.

Editor de trabalho visual

O editor visual de tarefas é uma interface gráfica que facilita a criação, a execução e o monitoramento de tarefas de extração, transformação e carregamento (ETL) em AWS Glue. Você pode compor visualmente fluxos de trabalho de transformação de dados e executá-los perfeitamente no mecanismo de ETL com tecnologia sem servidor no Apache Spark do AWS Glue e inspecionar o esquema e os resultados de dados em cada etapa do trabalho.

Operador

Com o AWS Glue, você só paga pelo tempo que seu trabalho de ETL leva para ser executado. Não há recursos para gerenciar, nenhum custo inicial e você não será cobrado pelo tempo de inicialização ou desligamento. É cobrada uma taxa por hora com base no número de unidades de processamento de dados (ou DPUs) usadas para executar seu trabalho de ETL. Uma única unidade de processamento de dados (DPU) também é chamada de operador. O AWS Glue vem com três tipos de operador para ajudá-lo a selecionar a configuração que atenda aos requisitos de latência e custo do trabalho. Os operadores vêm nas configurações Standard (Padrão), G.1X, G.2X e G.025X

Componentes do AWS Glue

O AWS Glue fornece um console e operações de API para configurar e gerenciar sua workload de extração, transformação e carregamento (ETL). Você pode usar operações de API por meio de vários SDKs específicos de linguagem e da AWS Command Line Interface (AWS CLI). Para obter informações sobre como usar a AWS CLI, consulte a Referência do comando da AWS CLI.

O AWS Glue usa o AWS Glue Data Catalog para armazenar metadados relacionado às fontes de dados, transformações e destinos. O Data Catalog é uma substituição inicial do Apache Hive Metastore. O AWS Glue Jobs system fornece uma infraestrutura gerenciada para definir, programar e executar operações de ETL nos seus dados. Para obter mais informações sobre a API do AWS Glue, consulte AWS Glue API.

Console do AWS Glue

O console do AWS Glue é usado para definir e orquestrar seu fluxo de trabalho de ETL. O console chama várias operações de API no AWS Glue Data Catalog e no AWS Glue Jobs system para executar as seguintes tarefas:

- Definir objetos do AWS Glue, como trabalhos, tabelas, crawlers e conexões.

- Programas quando os crawlers serão executados.
- Definir eventos ou programações para gatilhos de trabalho.
- Pesquisar e filtrar listas de objetos do AWS Glue.
- Editar scripts de transformação.

AWS Glue Data Catalog

O AWS Glue Data Catalog é o armazenamento de metadados técnicos persistentes na Cloud AWS.

Cada conta da AWS tem um AWS Glue Data Catalog por região da AWS. Cada Data Catalog é uma coleção altamente escalável de tabelas organizadas em bancos de dados. Uma tabela é uma representação de metadados de uma coleção de dados estruturados ou semiestruturados armazenados em fontes como Amazon RDS, Apache Hadoop Distributed File System, Amazon OpenSearch Service e outros. O AWS Glue Data Catalog fornece um repositório uniforme onde sistemas diferentes podem armazenar e encontrar metadados para acompanhar os dados em silos de dados. Você pode usar os metadados para consultar e transformar esses dados de maneira consistente em uma ampla variedade de aplicativos.

Você usa o Catálogo de Dados junto com as políticas AWS Identity and Access Management e o Lake Formation para controlar o acesso às tabelas e bancos de dados. Ao fazer isso, você pode permitir que diferentes grupos em sua empresa publiquem dados com segurança em toda a organização, protegendo informações confidenciais de maneira altamente granular.

O Data Catalog, juntamente com CloudTrail e Lake Formation, também fornece recursos abrangentes de auditoria e governança, com rastreamento de alterações de esquema e controles de acesso a dados. Isso ajuda a garantir que os dados não sejam modificados inadequadamente ou compartilhados inadvertidamente.

Para obter informações sobre como proteger e auditar o AWS Glue Data Catalog, consulte:

- AWS Lake Formation: para obter mais informações, consulte O que é o AWS Lake Formation? no Guia do desenvolvedor do AWS Lake Formation.
- CloudTrail — Para obter mais informações, consulte O que é CloudTrail? no Guia do usuário do AWS CloudTrail.

A seguir, estão outros produtos da AWS e projetos de código aberto que usam o AWS Glue Data Catalog:

- Amazon Athena: para obter mais informações, consulte Noções básicas de tabelas, bancos de dados e o Data Catalog no Manual do usuário do Amazon Athena.
- Amazon Redshift Spectrum: para obter mais informações, consulte Usar o Amazon Redshift Spectrum para consultar dados externos no Guia do desenvolvedor de banco de dados do Amazon Redshift.
- Amazon EMR: para obter mais informações, consulte Usar políticas com base em recursos para acesso do Amazon EMR ao AWS Glue Data Catalog no Guia de gerenciamento do Amazon EMR.
- Cliente do AWS Glue Data Catalog para o Apache Hive Metastore: para obter mais informações sobre esse projeto do GitHub, consulte Cliente do AWS Glue Data Catalog para o Apache Hive Metastore.

Crawlers e classificadores do AWS Glue

Com o AWS Glue, você também pode configurar os crawlers capazes de verificar dados em todos os tipos de repositórios, classificá-los, extrair informações de esquema deles e armazenar os metadados automaticamente no AWS Glue Data Catalog. O AWS Glue Data Catalog pode ser usado para guiar operações de ETL.

Para obter mais informações sobre como configurar crawlers e classificadores, consulte Usar crawlers para preencher o catálogo de dados . Para obter mais informações sobre como programar crawlers e classificadores usando a API do AWS Glue, consulte API de crawlers e classificadores.

Operações de ETL no AWS Glue

Ao usar os metadados no Data Catalog, o AWS Glue pode gerar automaticamente os scripts Scala ou PySpark (API do Python para Apache Spark) com extensões do AWS Glue que podem ser usadas e modificadas para executar várias operações de ETL. Por exemplo, você pode extrair, limpar e transformar dados brutos e, em seguida, armazenar o resultado em um repositório diferente onde ele poderá ser consultado e analisado. Esse script pode converter um arquivo CSV em um formulário relacional e salvá-lo no Amazon Redshift.

Para obter mais informações sobre como usar os recursos de ETL do AWS Glue, consulte Programar scripts do Spark.

ETL de streaming no AWS Glue

O AWS Glue permite executar operações de ETL em dados de transmissão usando trabalhos em execução contínua. O ETL de transmissão do AWS Glue é criado no mecanismo Apache Spark Structured Streaming e pode ingerir transmissões do Amazon Kinesis Data Streams, do Apache Kafka e do Amazon Managed Streaming for Apache Kafka (Amazon MSK). O ETL de transmissão pode limpar e transformar dados de transmissão e carregá-los no Amazon S3 ou em armazenamentos de dados JDBC. Use o ETL de streaming no AWS Glue para processar dados de eventos, como streams de IoT, streams de cliques e logs de rede.

Se você conhecer o esquema da fonte dos dados de transmissão, poderá especificá-lo em uma tabela do Data Catalog. Caso contrário, você pode habilitar a detecção de esquemas no trabalho de ETL de transmissão. Em seguida, o trabalho determina automaticamente o esquema dos dados recebidos.

O trabalho de ETL de transmissão pode usar tanto as transformações nativas do AWS Glue quanto as transformações nativas do Apache Spark Structured Streaming. Para obter mais informações, consulte Operations on streaming DataFrames/Datasets (Operações em transmissão de dataframes/conjuntos de dados) no site do Apache Spark.

Para ter mais informações, consulte the section called “Trabalhos de transmissão de ETL”.

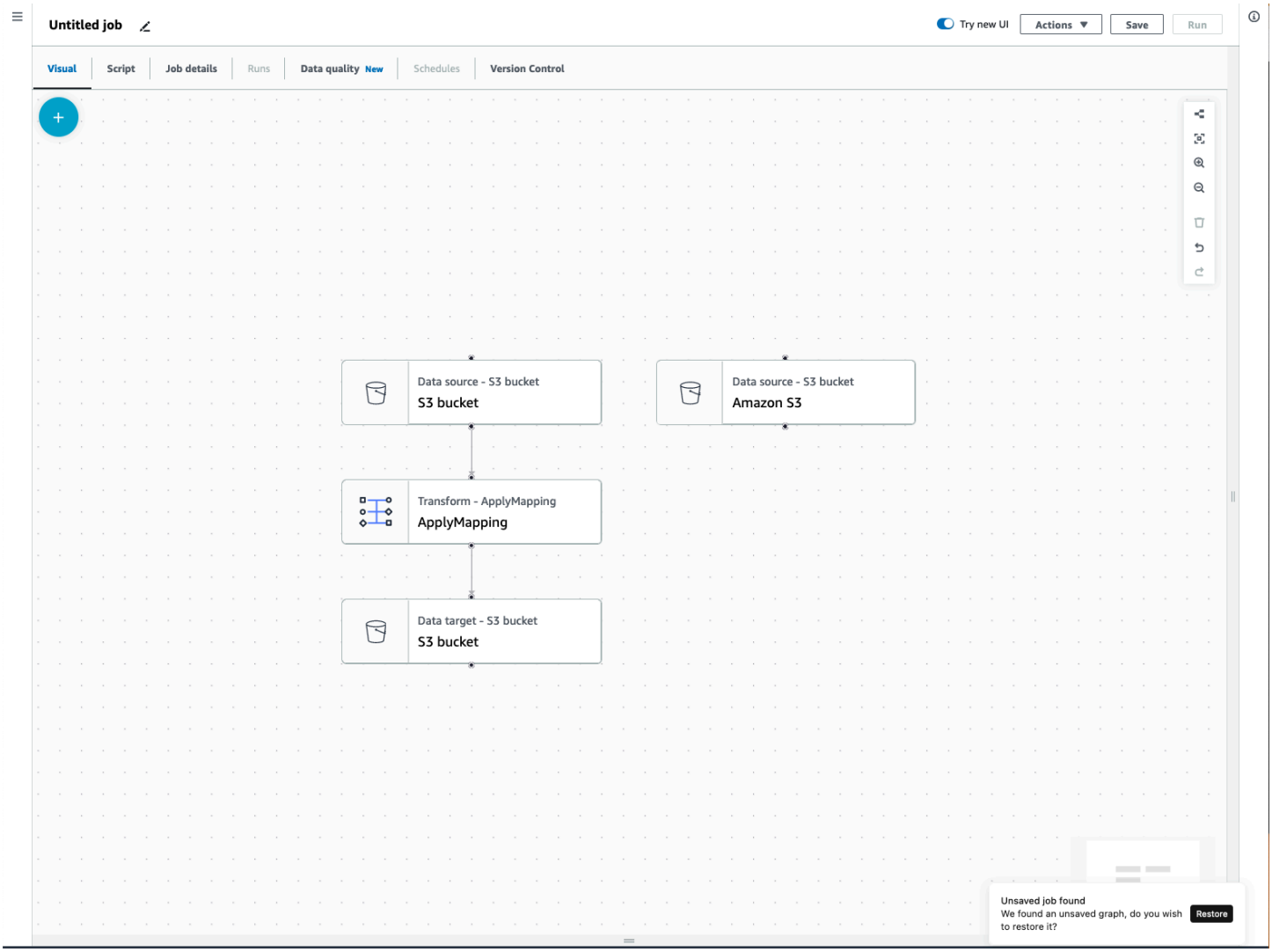
O sistema de trabalhos do AWS Glue

O AWS Glue Jobs system fornece infraestrutura gerenciada para orquestrar seu fluxo de trabalho de ETL. Você pode criar trabalhos no AWS Glue que automatizam os scripts usados para extrair, transformar e transferir dados para diferentes locais. Os trabalhos podem ser programados e encadeados, ou podem ser acionados por eventos como a chegada de novos dados.

Para obter mais informações sobre como usar o AWS Glue Jobs system, consulte Como monitorar o AWS Glue. Para obter informações sobre como programar usando a API do AWS Glue Jobs system, consulte API de trabalhos.

Componentes do Visual ETL

O AWS Glue permite criar trabalhos de ETL por meio de uma tela visual que você pode manipular.



Menu de trabalhos de ETL

As opções do menu na parte superior da tela permitem que você acesse as várias visualizações e detalhes de configuração sobre o seu trabalho.

- Visual: a tela do editor de trabalhos visual. Aqui, você pode adicionar nós para criar um trabalho.
- Script: a representação do script do seu trabalho de ETL. O AWS Glue gera o script com base na representação visual do seu trabalho. Você também pode editar seu script ou baixá-lo.

> **Note**
>
> Se você optar por editar o script, a experiência de criação do trabalho será permanentemente convertida em um modo somente de script. Depois disso, você não poderá mais usar o editor visual para editar o trabalho. Você deve adicionar todas as

fontes, transformações e destinos de trabalhos, e fazer todas as alterações necessárias com o editor visual antes de escolher editar o script.

- Detalhes do trabalho: a guia Detalhes do trabalho permite que você configure seu trabalho definindo as propriedades do trabalho. Há propriedades básicas, como nome e descrição do seu trabalho, perfil do IAM, tipo de trabalho, versão do AWS Glue, idioma, tipo de operador, número de operadores, marcador de trabalho, execução flexível, número de retiradas e tempo limite do trabalho, e há propriedades avançadas, como conexões, bibliotecas, parâmetros do trabalho e marcadores.
- Execuções: depois que seu trabalho for executado, essa guia poderá ser acessada para visualizar suas execuções de trabalhos anteriores.
- Qualidade dos dados: qualidade dos dados avalia e monitora a qualidade dos dados. Você pode aprender mais sobre como usar a qualidade de dados nessa guia e adicionar uma transformação de qualidade de dados ao seu trabalho.
- Agendamentos: os trabalhos que você agendou aparecem nessa guia. Se não houver agendamentos anexados a esse trabalho, essa guia não estará acessível.
- Controle de versão: você pode usar o Git com seu trabalho configurando seu trabalho em um repositório Git.

Painéis do Visual ETL

Quando você trabalha na tela, vários painéis estão disponíveis para ajudá-lo a configurar seus nós ou ajudá-lo a visualizar seus dados e visualizar o esquema de saída.

- Propriedades: o painel Propriedades aparece quando você escolhe um nó na tela.
- Visualização de dados: painel Visualização de dados fornece uma visualização prévia da saída de dados para que você possa tomar decisões antes de executar seu trabalho e examinar sua saída.
- Esquema de saída: a guia Esquema de saída permite que você visualize e edite o esquema dos seus nós de transformação.

Redimensionar painéis

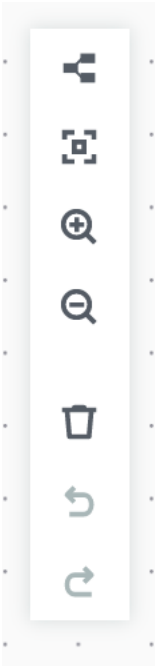
Você pode redimensionar o painel Propriedades no lado direito da tela e o painel inferior, que contém as guias Visualização de dados e Esquema de saída, clicando na borda do painel e arrastando-a para a esquerda e para a direita ou para cima e para baixo.

- Painel de propriedades: redimensione o painel de propriedades clicando e arrastando a borda da tela no lado direito da tela e arrastando-a para a esquerda para expandir sua largura. Por padrão, o painel é reduzido e, quando um nó é selecionado, o painel de propriedades se abre em seu tamanho padrão.
- Visualização de dados e painel Esquema de saída: redimensione o painel inferior clicando e arrastando a borda inferior na parte inferior da tela e arraste-a para cima para expandir sua altura. Por padrão, o painel é reduzido e, quando um nó é selecionado, o painel de inferior se abre em seu tamanho padrão.

Tela Trabalho

Você pode adicionar, remover e mover/reordenar nós diretamente na tela do Visual ETL. Pense nisso como seu espaço de trabalho para criar um trabalho de ETL totalmente funcional que começa com uma fonte de dados e pode terminar com um destino de dados.

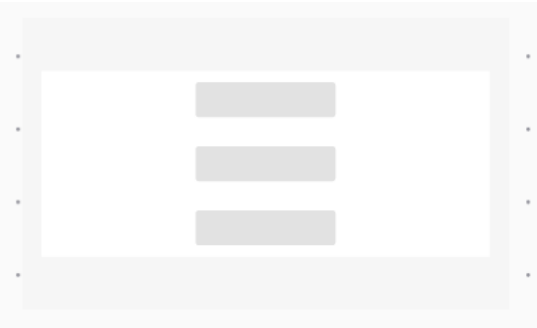
Ao trabalhar com nós na tela, você tem uma barra de ferramentas que pode ajudá-lo a ampliar e reduzir, remover nós, criar ou editar conexões entre nós, alterar a orientação do fluxo de trabalho e desfazer ou refazer uma ação.



A barra de ferramentas flutuante está ancorada no canto superior direito da tela e contém várias imagens que realizam ações:

- Ícone de layout: o primeiro ícone na barra de ferramentas é o ícone de layout. Por padrão, a direção dos trabalhos visuais é de cima para baixo. Ele reorganiza a direção do seu trabalho visual organizando os nós horizontalmente da esquerda para a direita. Clicar novamente no ícone do layout altera a direção de cima para baixo.
- Ícone de recentralização: o ícone de recentralização altera a visualização da tela ao centralizá-la. Você pode usar isso com trabalhos grandes para voltar à posição central.
- Ícone de ampliação de zoom: o ícone de ampliação de zoom aumenta o tamanho dos nós na tela.
- Ícone de redução de zoom: o ícone de redução de zoom reduz o tamanho dos nós na tela.
- Ícone da lixeira: o ícone de lixeira remove um nó de trabalho visual. Você deve primeiro selecionar um nó.
- Ícone de desfazer: o ícone de desfazer reverte a última ação realizada no trabalho visual.
- Ícone de refazer: o ícone de refazer repete a última ação realizada no trabalho visual.

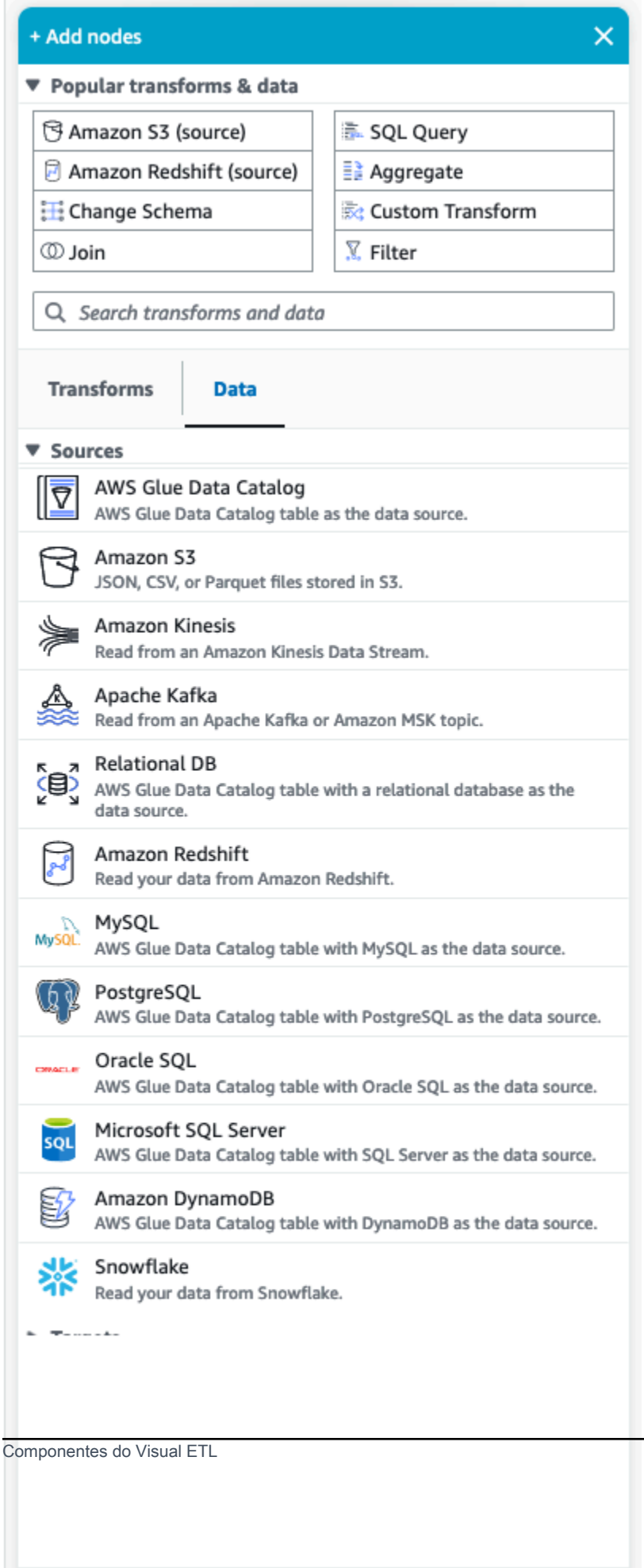
Usar o minimapa



Painel de recursos

O painel de recursos contém todas as fontes de dados, ações de transformação e conexões disponíveis para você. Abra o painel de recursos na tela clicando no ícone “+”. Isso abrirá o painel de recursos.

Para fechar o painel de recursos, clique no X no canto superior direito do painel de recursos. Isso ocultará o painel até que você esteja pronto para abri-lo novamente.


+ 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.

Transformações e dados populares

Na parte superior do painel, há uma coleção de transformações e dados populares. Esses nós são comumente usados no AWS Glue. Escolha um para adicioná-lo à tela. Você também pode ocultar as Transformações e dados populares clicando no triângulo ao lado do título Transformações e dados populares.

Abaixo da seção Transformações e dados populares, você pode pesquisar transformações e nós de fonte de dados. Os resultados aparecem à medida que você digita. Quanto mais letras você adicionar à sua consulta de pesquisa, menor ficará a lista de resultados. Os resultados da pesquisa são preenchidos a partir do nome e/ou descrição do nó. Escolha o nó para adicioná-lo à sua tela.

Transformações e dados

Há duas guias que organizam os nós em Transformações e Dados.

Transformações: quando você escolhe a guia Transformações, todas as transformações disponíveis podem ser selecionadas. Escolha uma transformação para adicioná-la à tela. Você também pode escolher Adicionar transformação na parte inferior da lista Transformações, o que abrirá uma nova página na documentação para criar Transformações visuais personalizadas. Seguir as etapas permitirá que você crie suas próprias transformações. Suas transformações aparecerão na lista de transformações disponíveis.

Dados: a guia de dados contém todos os nós de Fontes e Destinos. Você pode ocultar as Origens e os Destinos clicando no triângulo ao lado do título Origens ou Destinos. Você pode exibir as Origens e os Destinos clicando novamente no triângulo. Escolha um nó de origem ou de destino para adicioná-lo à tela. Você também pode escolher Gerenciar conexões para adicionar uma nova conexão. Isso abrirá a página Conectores no console.

AWS Glue para Spark e AWS Glue para Ray

No AWS Glue on Apache Spark (AWS Glue ETL), você pode usar o PySpark para escrever código do Python para lidar com dados em grande escala. O Spark é uma solução conhecida para esse problema, mas engenheiros de dados com experiência focada em Python podem achar que a transição não é intuitiva. O modelo DataFrame do Spark não é perfeitamente "phytônico", o que reflete a linguagem Scala e o runtime Java em que ele se baseia.

No AWS Glue, você pode usar trabalhos de shell do Python para executar integrações de dados nativos do Python. Esses trabalhos são executados em uma única instância do Amazon EC2 e são

limitados pela capacidade dessa instância. Isso restringe o throughput dos dados que você pode processar e sua manutenção se torna cara ao lidar com big data.

O AWS Glue para Ray permite que você aumente verticalmente a escala das workloads do Python sem investimentos substanciais para aprender o Spark. Você pode aproveitar certos cenários nos quais o Ray tem uma performance melhor. Oferecendo a você uma escolha, é possível usar os pontos fortes tanto do Spark quanto do Ray.

O AWS Glue ETL e o AWS Glue para Ray são diferentes internamente, portanto, são compatíveis com atributos diferentes. Verifique documentação para determinas quais são as configurações compatíveis.

O que é AWS Glue para Ray?

O Ray é uma estrutura de computação distribuída de código aberto que você pode usar para aumentar verticalmente a escala das workloads, com foco no Python. Para obter mais informações sobre o Ray, consulte o site do Ray. AWS Glue Os trabalhos e as sessões interativas do Ray permitem que você use o Ray no AWS Glue.

Você pode usar o AWS Glue para Ray para escrever scripts do Python para cálculos que serão executados em paralelo em várias máquinas. Em trabalhos e sessões interativas do Ray, você pode usar bibliotecas conhecidas do Python, como a pandas, para facilitar escrita e execução das workloads. Para obter mais informações sobre os conjuntos de dados do Ray, consulte Conjuntos de dados do Ray na documentação do Ray. Para obter mais informações sobre o pandas, consulte o site do pandas.

Ao usar o AWS Glue para Ray, você pode executar os fluxos de trabalho de pandas em big data em escala corporativa, com apenas algumas linhas de código. Você pode criar um trabalho do Ray no console do AWS Glue ou no AWS SDK. Você também pode abrir uma sessão interativa do AWS Glue para executar o código em um ambiente sem servidor do Ray. Trabalhos visuais no AWS Glue Studio ainda não são compatíveis.

Os trabalhos do AWS Glue para Ray permitem que você execute um script de acordo com uma agenda ou em resposta a um evento do Amazon EventBridge. As trabalhos armazenam informações de log e estatísticas de monitoramento no CloudWatch, as quais permitem que você entenda a integridade e a confiabilidade do seu script. Para obter mais informações sobre como usar o sistema de trabalhos do AWS Glue, consulte the section called “Trabalhar com trabalhos do Ray”.

As sessões interativas do AWS Glue para Ray (visualização) permitem que você execute trechos de código um após o outro nos mesmos recursos provisionados. Você pode usar isso para criar

protótipos e desenvolver scripts, com eficiência, ou para criar suas próprias aplicações interativas. Você pode usar as sessões interativas do AWS Glue de cadernos do AWS Glue Studio no AWS Management Console. Para obter mais informações, consulte Uso de cadernos com o AWS Glue Studio e o AWS Glue. Você também pode usá-los por meio de um kernel do Jupyter, que permite executar sessões interativas em ferramentas de edição de código existentes compatíveis com cadernos Jupyter, como o VSCode. Para ter mais informações, consulte the section called “AWS Glue para sessões interativas do Ray (pré-visualização)”.

O Ray automatiza o trabalho de escalação de código do Python distribuindo o processamento por um cluster de máquinas que ele reconfigura em tempo real, com base na carga. Isso pode resultar em uma melhor performance, em termos financeiros, para determinadas workloads. Com os trabalhos do Ray, incorporamos nativamente auto scaling ao modelo de trabalho do AWS Glue, para que você possa aproveitar totalmente esse recurso. Os trabalhos do Ray são executados no AWS Graviton, resultando em uma performance geral superior em termos de preço.

Além da economia de custos, você pode usar o auto scaling nativo para executar workloads do Ray sem investir tempo em manutenção, ajuste e administração de clusters. Você pode usar bibliotecas conhecidas de código aberto, prontas para uso, como pandas e AWS SDK for pandas. Isso melhora a velocidade de iteração enquanto você está desenvolvendo no AWS Glue para Ray. Ao usar o AWS Glue para Ray, você poderá desenvolver e executar rapidamente workloads de integração de dados com eficiência de custos.

Converter esquemas semi-estruturados em esquemas relacionais com o AWS Glue

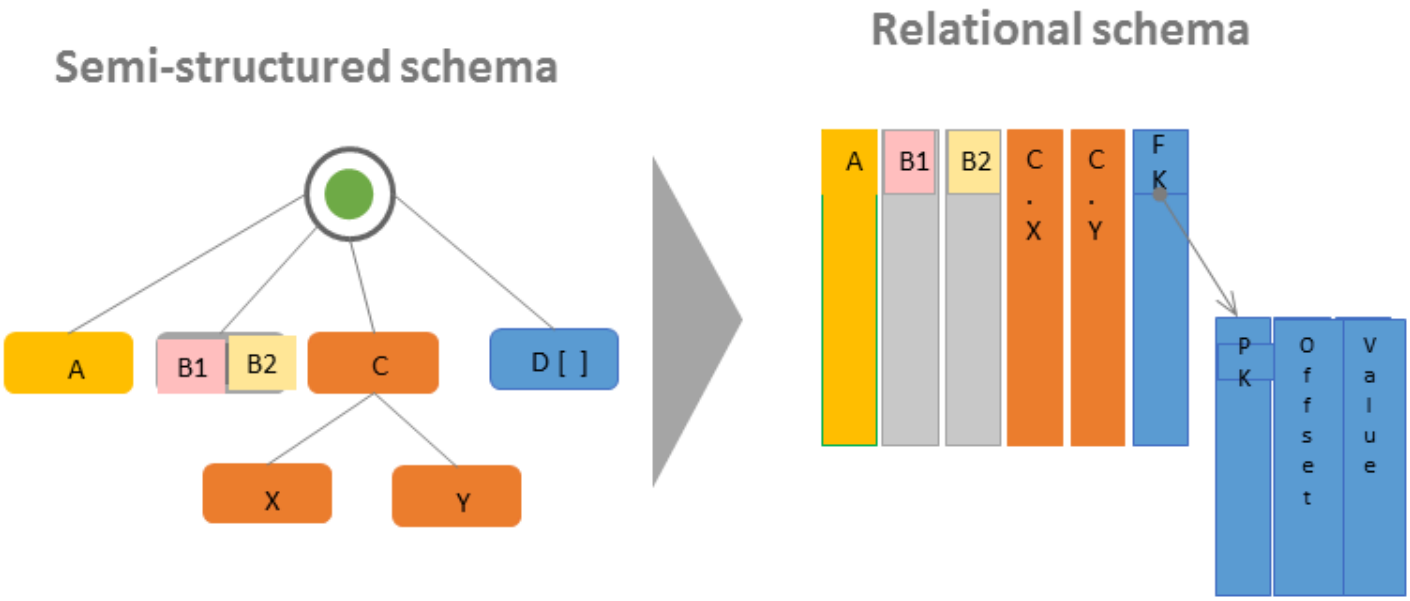
É comum querer converter dados semiestruturados em tabelas relacionais. Conceitualmente, você está nivelando um esquema hierárquico para um esquema relacional. O AWS Glue pode realizar essa conversão para você rapidamente.

Os dados semiestruturados normalmente contêm marcação para identificar entidades nos dados. Ele pode ter estruturas de dados aninhadas sem um esquema fixo. Para obter mais informações sobre dados semiestruturados, consulte Dados semiestruturados na Wikipédia.

Os dados relacionais são representados por tabelas que consistem em linhas e colunas. As relações entre tabelas podem ser representadas por uma relação de chave primária (PK) para chave externa (FK). Para obter mais informações, consulte Banco de dados relacional na Wikipédia.

O AWS Glue usa crawlers para inferir esquemas para dados semiestruturados. Em seguida, ele transforma os dados em um esquema relacional usando um trabalho de ETL (extração, transformação e carregamento). Por exemplo, você pode desejar analisar dados JSON desde os arquivos de origem do Amazon Simple Storage Service (Amazon S3) até as tabelas do Amazon Relational Database Service (Amazon RDS). Saber como o AWS Glue lida com as diferenças entre esquemas pode ajudar você a entender o processo de transformação.

Este diagrama mostra como o AWS Glue transforma um esquema semiestruturado em um esquema relacional.



O diagrama ilustra o seguinte:

- O valor único A converte-se diretamente em uma coluna relacional.
- O par de valores B1 e B2 se convertem em duas colunas relacionais.

- A estrutura `C`, com `X` e `Y` filhos, converte-se em duas colunas relacionais.
- A matriz `D[]` se converte em uma coluna relacional com uma chave externa (FK) que aponta para outra tabela relacional. Junto com uma chave primária (PK), a segunda tabela relacional apresenta colunas que contêm o deslocamento e o valor dos itens na matriz.

AWS Sistemas do tipo Glue

AWS O Glue usa sistemas de vários tipos para fornecer uma interface versátil em sistemas de dados que armazenam dados de maneiras muito diferentes. Este documento elimina a ambiguidade dos sistemas e padrões de dados do tipo AWS Glue.

AWS Tipos de catálogo de dados Glue

O Data Catalog é um registro de tabelas e campos armazenados em vários sistemas de dados, um metastore. Quando os componentes do AWS Glue, como AWS os rastreadores do AWS Glue e as tarefas do Glue with Spark, são gravados no Catálogo de Dados, eles o fazem com um sistema de tipos interno para rastrear os tipos de campos. Esses valores são mostrados na coluna Tipo de dados do esquema da tabela no AWS Glue Console. Esse sistema de tipos é baseado no sistema de tipos do Apache Hive. Para obter mais informações sobre o sistema de tipos do Apache Hive, consulte Tipos na wiki do Apache Hive. Para obter mais informações sobre tipos e suporte específicos, exemplos são fornecidos no AWS Glue Console, como parte do Schema Builder.

Validação, compatibilidade e outros usos

O Data Catalog não valida tipos gravados em campos de tipo. Quando os componentes do AWS Glue forem lidos e gravados no Catálogo de Dados, eles serão compatíveis entre si. AWS Os componentes do Glue também visam preservar um alto grau de compatibilidade com os tipos Hive. No entanto, os componentes do AWS Glue não garantem a compatibilidade com todos os tipos de Hive. Isso permite a interoperabilidade com ferramentas como o Athena DDL ao trabalhar com tabelas no Data Catalog.

Como o Data Catalog não valida tipos, outros serviços podem usar o Data Catalog para rastrear tipos usando sistemas estritamente conformes com o sistema do tipos do Hive ou qualquer outro sistema.

Tipos em AWS Glue com scripts Spark

Quando um script AWS Glue with Spark interpreta ou transforma um conjunto de dados, nós fornecemos `DynamicFrame` uma representação na memória do seu conjunto de dados conforme

ele é usado em seu script. O objetivo de um `DynamicFrame` é semelhante ao do `DataFrame` do Spark: ele modela o conjunto de dados para que o Spark possa agendar e executar transformações nos dados. Garantimos que a representação de tipo do `DynamicFrame` seja intercompatível com o `DataFrame`, fornecendo os métodos `toDF` e `fromDF`.

Se as informações de tipo puderem ser inferidas ou fornecidas a um `DataFrame`, elas poderão ser inferidas ou fornecidas a um `DynamicFrame`, a menos documentado o contrário. Quando fornecemos leitores ou gravadores otimizados para formatos de dados específicos, se o Spark puder ler ou gravar seus dados, nossos leitores e gravadores fornecidos poderão fazê-lo, sujeito às limitações documentadas. Para obter mais informações sobre leitores e gravadores, consulte the section called “Opções de formato de dados”.

O tipo Escolha

Os `DynamicFrames` fornecem um mecanismo para modelar campos em um conjunto de dados cujo valor pode ter tipos inconsistentes em disco entre as linhas. Por exemplo, um campo pode conter um número armazenado como uma string em determinadas linhas e um número inteiro em outras. Esse mecanismo é um tipo de memória denominado `Choice`. Fornecemos transformações, como o `ResolveChoice` método, para transformar as colunas Choice em um tipo concreto. AWS O Glue ETL não gravará o tipo Choice no Catálogo de Dados no curso normal da operação; os tipos de escolha só existem no contexto dos modelos de DynamicFrame memória dos conjuntos de dados. Para obter um exemplo do uso do tipo Escolha, consulte the section called “Exemplo de preparo de dados”.

AWS Tipos de Glue Crawler

Os rastreadores têm como objetivo produzir um esquema consistente e utilizável para seu conjunto de dados e, em seguida, armazená-lo no Catálogo de Dados para uso em outros componentes do AWS Glue e no Athena. Os crawlers lidam com os tipos conforme descrito na seção anterior sobre o Data Catalog, the section called “AWS Tipos de catálogo de dados Glue”. Para produzir cenários utilizáveis do tipo "Escolha", em que uma coluna contém valores de dois ou mais tipos, os crawlers criam um tipo de `struct` que modela os tipos possíveis.

Conceitos básicos do AWS Glue

As seções a seguir fornecem informações sobre como configurar o AWS Glue. Nem todas as seções de configuração são necessárias para começar a usar o AWS Glue. Você pode usar as instruções, conforme necessário, para configurar permissões do IAM, criptografia e DNS (se estiver usando um ambiente de VPC para acessar armazenamentos de dados ou se estiver usando sessões interativas).

Tópicos

Visão geral do uso do AWS Glue

Com o AWS Glue, você armazena metadados no AWS Glue Data Catalog. Você usa esses metadados para orquestrar trabalhos de ETL que transformam fontes de dados e carregam o data warehouse ou o data lake. As etapas a seguir descrevem o fluxo de trabalho geral e algumas das opções que você faz ao trabalhar com o AWS Glue.

> **Note**
>
> É possível realizar as etapas a seguir ou criar um fluxo de trabalho que execute automaticamente as etapas 1 a 3. Para ter mais informações, consulte the section called "Realizar atividades de ETL complexas usando esquemas e fluxos de trabalho".

1. Preencher o AWS Glue Data Catalog com definições de tabela.

   No console, para armazenamentos de dados persistentes, é possível adicionar um crawler para preencher o AWS Glue Data Catalog. Você pode iniciar o assistente Add crawler na lista de

tabelas ou na lista de crawlers. Você escolhe um ou mais armazenamentos de dados para o seu crawler acessar. Você também pode criar uma programação para determinar a frequência de execução do seu crawler. Para streams de dados, é possível criar manualmente a definição de tabela e definir propriedades de stream.

Se preferir, você pode fornecer um classificador personalizado que infere o esquema dos seus dados. Você pode criar classificadores personalizados usando um padrão grok, No entanto, o AWS Glue fornece classificadores integrados que são usados automaticamente pelos crawlers quando um classificador personalizado não reconhece seus dados. Ao definir um crawler, você não precisa selecionar um classificador. Para obter mais informações sobre classificadores no AWS Glue, consulte Adicionar classificadores a um crawler no AWS Glue.

O crawling de alguns tipos de armazenamento de dados requer uma conexão que forneça informações de autenticação e local. Se necessário, você pode criar uma conexão que fornece essas informações necessárias no console do AWS Glue.

O crawler lê seu armazenamento de dados e cria definições de dados e tabelas nomeadas no AWS Glue Data Catalog. Estas tabelas são organizadas em um banco de dados de sua escolha. Você também pode preencher o Data Catalog com tabelas criadas manualmente. Com esse método, você fornece o esquema e outros metadados para criar definições de tabela no Data Catalog. Como esse método pode ser um pouco entediante e propenso a erros, muitas vezes é melhor deixar que o crawler crie as definições da tabela.

Para obter mais informações sobre como preencher o AWS Glue Data Catalog com definições de tabela, consulte Criar tabelas.

2. Defina um trabalho que descreva a transformação de dados de origem para dados de destino.

   Geralmente, para criar um trabalho, você precisa fazer as seguintes escolhas:

   - Escolha uma tabela do AWS Glue Data Catalog para ser a origem do trabalho. Seu trabalho usa essa definição de tabela para acessar a fonte de dados e interpretar o formato dos dados.
   - Escolha uma tabela ou um local do AWS Glue Data Catalog para ser o destino do trabalho. Seu trabalho usa essas informações para acessar seu armazenamento de dados.
   - Instrua o AWS Glue a gerar um script para transformar sua origem em destino. O AWS Glue gera o código para chamar transformações integradas a fim de converter dados de seu esquema de origem no formato do esquema de destino. Essas transformações executam operações como cópia de dados, renomeação de colunas e filtragem de dados para transformar os dados conforme necessário. Você pode modificar esse script no console do AWS Glue.

Para obter mais informações sobre como definir trabalhos no AWS Glue, consulte Criar trabalhos ETL visuais com o AWS Glue Studio.

3. Execute seu trabalho para transformar seus dados.

   Você pode executar seu trabalho sob demanda, ou iniciá-lo com base em um destes tipos de gatilho:

   - Um gatilho baseado em uma programação cron.
   - Um gatilho baseado em eventos. Por exemplo, a conclusão bem-sucedida de outro trabalho pode iniciar um trabalho do AWS Glue.
   - Um gatilho que inicia um trabalho sob demanda.

   Para obter mais informações sobre gatilhos no AWS Glue, consulte Iniciar trabalhos e crawlers usando acionadores.

4. Monitore seus crawlers programados e trabalhos acionados.

   Use o console do AWS Glue para visualizar o seguinte:

   - Detalhes e erros de execução do trabalho.
   - Detalhes e erros de execução do crawler.
   - Todas as notificações sobre atividades do AWS Glue

   Para obter mais informações sobre como monitorar crawlers e trabalhos no AWS Glue, consulte Como monitorar o AWS Glue.

Configurar permissões do IAM para o AWS Glue

As instruções neste tópico ajudam você a configurar rapidamente as permissões AWS Identity and Access Management (IAM) para o AWS Glue. Você concluirá as seguintes tarefas:

- Conceda às suas identidades do IAM acesso aos recursos do AWS Glue.
- Crie um perfil de serviço para executar trabalhos, acessar dados e executar tarefas de qualidade do AWS Glue Data Quality.

Para obter instruções detalhadas que você pode usar para personalizar as permissões do IAM para o AWS Glue, consulte Configurar permissões do KMS para o AWS Glue.

Para configurar as permissões do IAM para AWS Glue no AWS Management Console

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Escolha Conceitos básicos.
3. Em Preparar sua conta para o AWS Glue, escolha Configurar permissões do IAM.
4. Escolha as identidades do IAM (perfis ou usuários) às quais você deseja conceder permissões do AWS Glue. O AWS Glue anexa a política gerenciada `AWSGlueConsoleFullAccess` a essas identidades. Você pode pular essa etapa se quiser definir essas permissões manualmente ou se quiser definir apenas um perfil de serviço padrão.
5. Escolha Próximo.
6. Escolha o nível de acesso ao Amazon S3 de que os perfis e usuários precisam. As opções escolhidas nessa etapa são aplicadas a todas as identidades selecionadas.

    a. Em Escolher locais do S3, escolha os locais do Amazon S3 aos quais você deseja conceder acesso.

    b. Em seguida, selecione se as identidades devem ter acesso de Somente leitura (recomendado) ou de Leitura e gravação aos locais que você selecionou anteriormente. O AWS Glue adiciona políticas de permissões às identidades com base na combinação dos locais e das permissões de leitura ou gravação que você seleciona.

    A tabela a seguir mostra as permissões que o AWS Glue anexa para acesso ao Amazon S3.

Se você escolher...	O AWS Glue anexará...
Nenhuma alteração	Nenhuma alteração. O AWS Glue não fará nenhuma alteração nas permissões da sua identidade.
Conceder acesso a locais específicos do Amazon S3 (somente leitura)	Uma política em linha incorporada nas identidades do IAM selecionadas. Para obter mais informações, consulte Políticas em linha no Guia do usuário do IAM.

Se você escolher...	O AWS Glue anexará...
	O AWS Glue atribui um nome à política usando a seguinte convenção : `AWSGlueConsole <Role/User> InlinePolicy-read-specific-access- <UUID>`. Por exemplo: `AWSGlueConsoleRole InlinePolicy-read-specific-access-123456780123` . Este é exemplo de uma política em linha que o AWS Glue anexa para conceder acesso somente leitura a um local específico do Amazon S3. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "s3:Get*", "s3:List*" ], "Resource": [ "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*" ] } ] }`

Se você escolher...	O AWS Glue anexará...
Conceder acesso a locais específicos do Amazon S3 (leitura e gravação)	Uma política em linha incorporada nas identidades do IAM selecionadas. Para obter mais informações, consulte Políticas em linha no Guia do usuário do IAM. O AWS Glue atribui um nome à política usando a seguinte convenção : `AWSGlueConsole` *`<Role/User>`* `InlinePolicy-read -and-write-specific-access-` *`<UUID>`*. Por exemplo: `AWSGlueConsoleRole InlinePolicy-read-and-write-specific-access-123456780123 .` Este é exemplo de uma política em linha que é AWS Glue anexada para conceder acesso de leitura e gravação a locais específicos do 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/*" ] }`

Se você escolher...	O AWS Glue anexará...
	] }
Conceder total acesso ao Amazon S3 (somente leitura)	A política do IAM gerenciada `AmazonS3ReadOnlyAccess`. Para saber mais, consulte a política gerenciada pela AWS: AmazonS3ReadOnlyAccess.
Conceder total acesso ao Amazon S3 (leitura e gravação)	A política do IAM gerenciada `AmazonS3FullAccess`. Para saber mais, consulte a política gerenciada pela AWS: AmazonS3FullAccess.

7. Escolha Próximo.
8. Escolha perfil de serviço padrão do AWS Glue para sua conta. Um perfil de serviço é um perfil do IAM que é o AWS Glue usada para acessar recursos em outros serviços da AWS em seu nome. Para ter mais informações, consulte Perfis de serviço do AWS Glue.

   - Ao escolher perfil de serviço padrão do AWS Glue, o AWS Glue cria um novo perfil do IAM na sua Conta da AWS denominado `AWSGlueServiceRole` com as seguintes políticas gerenciadas anexadas. Se sua conta já tiver um perfil do IAM denomiado `AWSGlueServiceRole`, o AWS Glue anexa essas políticas ao perfil existente.
     - AWSGlueServiceRole
     - AmazonS3FullAccess
   - Quando você escolhe um perfil do IAM existente, o AWS Glue o define como padrão, mas não adiciona a ele nenhuma permissão. Verifique se você configurou o perfil a ser usado como um perfil de serviço do AWS Glue. Para obter mais informações, consulte Etapa 1: criar uma política do IAM para o serviço AWS Glue e Etapa 2: criar um perfil do IAM para o AWS Glue.

9. Escolha Próximo.
10. Por fim, revise as permissões que você selecionou e escolha Aplicar alterações. Ao aplicar as alterações, o AWS Glue adiciona permissões do IAM às identidades que você selecionou. Você pode visualizar ou modificar as novas permissões no console do IAM em https://console.aws.amazon.com/iam/.

Você agora concluiu a configuração de permissões mínimas do IAM para o AWS Glue. Em um ambiente de produção, recomendamos que você se familiarize com o Segurança no AWS Glue e o Gerenciamento de identidade e acesso para AWS Glue para ajudar a proteger os recursos da AWS para seu caso de uso.

Próximas etapas

Agora que tem as permissões do IAM configuradas, você pode explorar os seguintes tópicos para começar a usar o AWS Glue:

- Introdução ao AWS Glue no AWS Skill Builder
- Conceitos básicos da AWS Glue Data Catalog

Configuração do AWS Glue Studio

Conclua as tarefas nesta seção ao usar o AWS Glue para o Visual ETL pela primeira vez:

Tópicos

- Revisar as permissões do IAM necessárias para o usuário do AWS Glue Studio
- Revisar as permissões do IAM necessárias para trabalhos de ETL
- Configurar permissões do IAM para o AWS Glue Studio
- Configurar uma VPC para seu trabalho de ETL

Revisar as permissões do IAM necessárias para o usuário do AWS Glue Studio

Para usar AWS Glue Studio, o usuário deve ter acesso a vários recursos da AWS. O usuário deve ser capaz de visualizar e selecionar buckets do Amazon S3, políticas e funções do IAM e objetos do AWS Glue Data Catalog.

Permissões de serviço do AWS Glue

O AWS Glue Studio usa as ações e recursos do serviço do AWS Glue. Seu usuário precisa de permissões sobre essas ações e recursos para usar com eficiência o AWS Glue Studio. Você pode conceder ao usuário do AWS Glue Studio a política gerenciada `AWSGlueConsoleFullAccess` ou criar uma política personalizada com um conjunto menor de permissões.

> **Important**
>
> De acordo com as práticas recomendadas de segurança, recomenda-se restringir o acesso enrijecendo políticas para restringir ainda mais o acesso ao bucket do Amazon S3 e grupos de logs do Amazon CloudWatch. Para obter um exemplo de política do Amazon S3, consulte Gravar políticas do IAM: como conceder acesso a um bucket do Amazon S3.

Criação de políticas do IAM personalizadas para o AWS Glue Studio

Você pode criar uma política personalizada com um conjunto menor de permissões para o AWS Glue Studio. A política pode conceder permissões para um subconjunto de objetos ou ações. Use as seguintes informações ao criar uma política personalizada.

Para usar as APIs do AWS Glue Studio, inclua `glue:UseGlueStudio` na política de ação em suas permissões do IAM. O uso de `glue:UseGlueStudio` permitirá que você acesse todas as ações do AWS Glue Studio mesmo quando mais ações forem adicionadas à API no decorrer do tempo.

Ações do gráfico acíclico dirigido (DAG)

- CreateDag
- UpdateDag
- GetDag
- DeleteDag

Ações de trabalho

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

Ações de execução de trabalho

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

Ações do esquema

- GetSchema
- GetInferredSchema

Ações de banco de dados

- GetDatabases

Ações de planejamento

- GetPlan

Ações de tabela

- SearchTables
- GetTables
- GetTables

Ações de conexão

- CreateConnection
- DeleteConnection
- UpdateConnection
- GetConnections
- GetConnection

Ações de mapeamento

- GetMapping

Ações do proxy do S3

- ListBuckets
- ListObjectsV2
- GetBucketLocation

Ações de configuração de segurança

- GetSecurityConfigurations

Ações de script

- CreateScript (diferente da API de mesmo nome no AWS Glue)

Acesso às APIs do AWS Glue Studio

Para acessar o AWS Glue Studio, adicione `glue:UseGlueStudio` na lista de políticas de ações nas permissões do IAM.

No exemplo abaixo, `glue:UseGlueStudio` está incluído na política de ação, mas as APIs do AWS Glue Studio não estão identificadas individualmente. Isso ocorre porque ao incluir `glue:UseGlueStudio`, você recebe automaticamente acesso às APIs internas sem precisar especificar as APIs individuais do AWS Glue Studio nas permissões do IAM.

No exemplo, as políticas adicionais de ação listadas (p. ex., `glue:SearchTables`) não são APIs do AWS Glue Studio, portanto, será necessário incluí-las nas permissões do IAM, conforme necessário. Você também pode incluir ações do Amazon S3 Proxy para especificar o nível de acesso do Amazon S3 a ser concedido. O exemplo de política abaixo fornece acesso para abrir o AWS Glue Studio, criar um trabalho visual e salvá-lo/executá-lo se o perfil do IAM selecionado tiver acesso suficiente.

```
{
    "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"
                    ]
                }
            }
        }
    ]
}
```

Permissões de visualização de dados e caderno

As visualizações de dados e os cadernos permitem que você veja uma amostra dos seus dados em qualquer estágio do trabalho (leitura, transformação, gravação), sem precisar executar o trabalho. Você especifica uma função do AWS Identity and Access Management (IAM) para a AWS Glue Studio para usar ao acessar os dados. As funções do IAM destinam-se a poder ser assumidas e não têm credenciais de longo prazo padrão, como uma senha ou chaves de acesso, associadas a ela. Em vez disso, quando o AWS Glue Studio assume a função, o IAM fornece credenciais de segurança temporárias.

Para garantir que as visualizações de dados e os comandos do caderno funcionem corretamente, use uma função que tenha um nome que comece com a string `AWSGlueServiceRole`. Se você escolher usar um nome diferente para a função, será necessário adicionar a permissão `iam:passrole` e configurar uma política para a função no IAM. Para ter mais informações, consulte Criar uma política do IAM para funções não nomeadas “AWSGlueServiceRole*”.

> **Warning**
>
> Se uma função conceder a permissão `iam:passrole` a um caderno e você implementar o encadeamento de funções, um usuário pode inadvertidamente obter acesso ao caderno. No momento, não há auditoria implementada que permita monitorar quais usuários receberam acesso ao caderno.

Caso queira negar a uma identidade do IAM a capacidade de criar sessões de pré-visualização de dados, consulte o exemplo para the section called “Negação da capacidade de criar sessões de pré-visualização de dados para uma identidade”.

Permissões Amazon CloudWatch

É possível monitorar os trabalhos do AWS Glue Studio usando o Amazon CloudWatch, o qual coleta e processa dados brutos do AWS Glue e os transforma em métricas legíveis quase em tempo real. Por padrão, os dados de métricas do AWS Glue são enviados para o CloudWatch automaticamente. Para obter mais informações, consulte O que é o Amazon CloudWatch? no Manual do usuário do Amazon CloudWatch e Métricas do AWS Glue no Guia do desenvolvedor do AWS Glue.

Para acessar os painéis do CloudWatch, o usuário que acessa o AWS Glue Studio precisa de um dos seguintes:

- A política `AdministratorAccess`
- A política `CloudWatchFullAccess`
- Uma política personalizada que inclui uma ou mais destas permissões específicas:
  - `cloudwatch:GetDashboard` e `cloudwatch:ListDashboards` para visualizar painéis
  - `cloudwatch:PutDashboard` para criar ou modificar painéis
  - `cloudwatch:DeleteDashboards` para excluir painéis

Para obter mais informações sobre como alterar as permissões de um usuário do IAM usando políticas, consulte Alterar permissões de um usuário do IAM no Manual do usuário do IAM.

Revisar as permissões do IAM necessárias para trabalhos de ETL

Quando você cria um trabalho usando o AWS Glue Studio, o trabalho assume as permissões da função do IAM que você especificou ao criá-lo. Essa função do IAM precisa ter permissão para extrair dados da sua origem dos dados, gravá-los no destino e acessar recursos do AWS Glue.

O nome da função que você cria para o trabalho deve começar com a string `AWSGlueServiceRole` para que ele seja usado corretamente pelo AWS Glue Studio. Por exemplo, você pode nomear sua função `AWSGlueServiceRole-FlightDataJob`.

Permissões de origem e destino dos dados

Um trabalho do AWS Glue Studio deve ter acesso ao Amazon S3 para quaisquer origens, destinos, scripts e diretórios temporários utilizados no trabalho. Você pode criar uma política para fornecer acesso minucioso a recursos específicos do Amazon S3.

- As fontes de dados exigem permissões `s3:ListBucket` e `s3:GetObject`.
- Os destinos de dados exigem permissões `s3:ListBucket`, `s3:PutObject` e `s3:DeleteObject`.

Se escolher Amazon Redshift como sua origem dos dados, você poderá fornecer uma função para permissões de cluster. Trabalhos que são executados em um cluster do Amazon Redshift emitem comandos que acessam o Amazon S3 para armazenamento temporário usando credenciais temporárias. Se o trabalho for executado por mais de uma hora, essas credenciais expirarão, fazendo com que o trabalho falhe. Para evitar esse problema, você pode atribuir uma função ao próprio cluster do Amazon Redshift que concede as permissões necessárias aos trabalhos utilizando credenciais temporárias. Para obter mais informações, consulte Mover dados de e para o Amazon RedShift no Guia do desenvolvedor do AWS Glue.

Se o trabalho usar origens ou destinos de dados diferentes do Amazon S3, você deve anexar as permissões necessárias à função do IAM usada pelo trabalho para acessar essas origens e destinos de dados. Para obter mais informações, consulte Configurar seu ambiente para acessar armazenamentos de dados no Guia do desenvolvedor do AWS Glue.

Se você estiver usando conectores e conexões para seu armazenamento de dados, precisará de permissões adicionais, conforme descrito em the section called “Permissões necessárias para usar conectores”.

Permissões necessárias para excluir trabalhos

No AWS Glue Studio, você pode selecionar vários trabalhos no console para excluir. Para executar essa ação, você deve ter a permissão `glue:BatchDeleteJob`. Isso é diferente do console do AWS Glue, que requer a permissão `glue:DeleteJob` para excluir trabalhos.

Permissões AWS Key Management Service

Se você planeja acessar origens e destinos do Amazon S3 que usam criptografia do lado do servidor com AWS Key Management Service (AWS KMS), anexe uma política à função do AWS Glue Studio usada pelo trabalho que permita que ele descriptografe os dados. A função de trabalho precisa

das permissões `kms:ReEncrypt`, `kms:GenerateDataKey` e `kms:DescribeKey`. Além disso, a função de trabalho precisa da permissão `kms:Decrypt` para carregar ou baixar um objeto do Amazon S3 criptografado com uma chave mestra do cliente do AWS KMS (CMK).

Há custos adicionais pelo uso de CMKs do AWS KMS. Para obter mais informações, consulte Conceitos do AWS Key Management Service: chaves mestras de cliente (CMKs) em Preços do AWS Key Management Service no Guia do desenvolvedor do AWS Key Management Service.

Permissões necessárias para usar conectores

Se você estiver usando um conector personalizado e uma conexão do AWS Glue para acessar um armazenamento de dados, a função usada para executar o trabalho de ETL do AWS Glue precisa de permissões adicionais anexadas:

- A política gerenciada pela AWS `AmazonEC2ContainerRegistryReadOnly` para acessar conectores comprados no AWS Marketplace.
- As permissões `glue:GetJob` e `glue:GetJobs`.
- Permissões do AWS Secrets Manager para acessar segredos que são usados com conexões. Consulte Exemplo: permissão para recuperar valores de segredos para obter exemplos de políticas do IAM.

Se seus trabalhos de ETL do AWS Glue são executados em uma VPC executando a Amazon VPC, então a VPC deve ser configurada conforme descrito em the section called “Configurar uma VPC para seu trabalho de ETL”.

Configurar permissões do IAM para o AWS Glue Studio

Você pode criar as funções e atribuir políticas a usuários e a funções de trabalho usando o usuário administrador da AWS.

Você pode usar a política gerenciada pela AWS AWSGlueConsolefullAccess para fornecer as permissões necessárias para usar o console do AWS Glue Studio.

Para criar sua própria política, siga as etapas documentadas em Criar uma política do IAM para o serviço do AWS Glue no Guia do desenvolvedor do AWS Glue. Inclua as permissões do IAM descritas anteriormente em Revisar as permissões do IAM necessárias para o usuário do AWS Glue Studio.

Tópicos

- Anexar políticas ao usuário do AWS Glue Studio
- Criar uma política do IAM para funções não nomeadas “AWSGlueServiceRole*”

Anexar políticas ao usuário do AWS Glue Studio

Qualquer usuário do AWS que fizer login no console do AWS Glue Studio deverá ter permissões para acessar recursos específicos. Você fornece essas permissões associando políticas do IAM ao usuário.

Para anexar a política gerenciada AWSGlueConsoleFullAccess a um usuário

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Políticas.
3. Na lista de políticas, marque a caixa de seleção ao lado da política AWSGlueConsoleFullAccess. Você pode usar o menu Filtro e a caixa de pesquisa para filtrar a lista de políticas.
4. Selecione Ações da política e escolha Anexar.
5. Escolha o usuário ao qual a política será anexada. Você pode usar o menu Filter (Filtro) e a caixa de pesquisa para filtrar a lista de entidades principais. Depois de escolher o usuário ao qual a política será anexada, selecione Attach policy.
6. Repita as etapas anteriores para anexar políticas adicionais ao usuário conforme necessário.

Criar uma política do IAM para funções não nomeadas “AWSGlueServiceRole*”

Para configurar uma política do IAM para funções usadas pelo AWS Glue Studio

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Adicione uma nova política do IAM. É possível adicionar a uma política existente ou criar uma nova política em linha do IAM. Para criar uma política do IAM:
    1. Escolha Policies (Políticas) e depois Create Policy (Criar política). Se aparecer um botão Get Started, selecione-o e, em seguida, Create Policy.
    2. Próximo a Create Your Own Policy, escolha Select.
    3. Em Nome da política, digite qualquer valor que seja mais fácil para você consultar mais tarde. Você também pode digitar um texto descritivo em Descrição.

4. Em Documento de política, digite uma instrução com o seguinte formato e escolha Criar política:

3. Copie e cole os blocos a seguir na política na matriz “Statement”, substituindo *my-interactive-session-role-prefix* pelo prefixo de todos os perfis comuns a serem associados com as permissões para o AWS Glue.

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

Aqui está o exemplo completo com os arrays Version and Statement incluídos na 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 a política para um usuário, escolha Usuários.
5. Escolha o usuário do para o qual você deseja anexar a política.

Configurar uma VPC para seu trabalho de ETL

O Amazon Virtual Private Cloud (Amazon VPC) permite definir uma rede virtual em sua própria área isolada logicamente na Nuvem AWS, conhecida como uma nuvem privada virtual (VPC). Você pode iniciar os recursos da AWS, como as instâncias, na VPC. Sua VPC assemelha-se a uma rede tradicional que você poderia operar no seu próprio datacenter, com os benefícios de usar a infraestrutura escalável da AWS. É possível configurar seu VPC, selecionar o intervalo de endereços IP dele, criar sub-redes e definir tabelas de rotas, gateways de rede e configurações de segurança. Você pode se conectar às suas instâncias na VPC. É possível conectar a VPC a seu próprio datacenter corporativo, tornando a Nuvem AWS uma extensão do seu datacenter. Para proteger os recursos em cada sub-rede, você pode usar várias camadas de segurança, incluindo grupos de segurança e listas de controle de acesso de rede. Para obter mais informações, consulte o Manual do usuário da Amazon VPC.

Você pode configurar os trabalhos de ETL do AWS Glue para serem executados em uma VPC ao usar conectores. Você deve configurar a VPC para o seguinte, conforme necessário:

- Acesso à rede pública para armazenamentos de dados fora da AWS. Todos os armazenamentos de dados acessados pelo trabalho devem estar disponíveis na sub-rede da VPC.
- Se o seu trabalho precisar acessar recursos da VPC e a Internet pública, a VPC precisará conter uma instância de gateway de conversão de endereços de rede (NAT).

  Para obter mais informações, consulte Configurar seu ambiente para acessar armazenamentos de dados no Guia do desenvolvedor do AWS Glue.

Conceitos básicos de cadernos no AWS Glue Studio

Ao iniciar um caderno por meio do AWS Glue Studio, todas as etapas de configuração são feitas para você, para que você possa explorar seus dados e começar a desenvolver seu script de trabalho após apenas alguns segundos.

As seções a seguir descrevem como criar um perfil e conceder as permissões apropriadas para usar cadernos do AWS Glue Studio para trabalhos de ETL.

Tópicos

Conceder permissões para a função do IAM

A configuração do AWS Glue Studio é um pré-requisito para usar cadernos.

Para usar cadernos no AWS Glue, seu perfil exige o seguinte:

- Uma relação de confiança com o AWS Glue para a ação `sts:AssumeRole`, além de `sts:TagSession` se você quiser usar marcação.
- Uma política do IAM contendo todas as operações de API para cadernos,AWS Glue e sessões interativas.
- Uma política do IAM para uma função de transmissão, pois a função precisa ser capaz de repassar ela própria do caderno para sessões interativas.

Por exemplo, ao criar um novo perfil, você pode adicionar uma política gerenciada padrão do AWS como `AWSGlueConsoleFullAccessRole` a função, e depois adicionar uma nova política para as operações do caderno e outra para a política PassRole do IAM.

Ações necessárias para uma relação de confiança com o AWS Glue

Ao iniciar uma sessão de caderno, deve adicionar `sts:AssumeRole` ao relacionamento de confiança da função que é repassada ao caderno. Se a sessão incluir etiquetas, você também deve repassar a ação `sts:TagSession`. Sem essas ações, a sessão do caderno não pode começar.

Por exemplo:

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

```
    ]
}
```

Políticas que contêm as operações de API para cadernos

O exemplo de política a seguir descreve as permissões necessárias do AWS IAM para os cadernos. Se você estiver criando um novo perfil, crie uma política que contenha o seguinte:

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

É possível usar as seguintes políticas do IAM para permitir acesso a recursos específicos:

- AwsGlueSessionUserRestrictedNotebookServiceRole: fornece acesso total a todos os recursos do AWS Glue, exceto para sessões. Permite que os usuários criem e usem somente as sessões de caderno que estejam associadas ao usuário. Essa política também inclui outras permissões exigidas pelo AWS Glue para gerenciar recursos do AWS Glue em outros serviços da AWS.
- AwsGlueSessionUserRestrictedNotebookPolicy: fornece permissões que permitem que os usuários criem e usem somente as sessões de caderno associadas ao usuário. Essa política também inclui permissões para permitir explicitamente que os usuários transmitam uma função de sessão restrita do AWS Glue.

Política do IAM para uma aprovar um perfil

Quando você cria um caderno com uma função, essa função é transmitida para sessões interativas, de modo que a mesma função possa ser usada em ambos os lugares. Por isso, a permissão `iam:PassRole` precisa fazer parte da política da função.

Crie uma nova política para seu perfil usando o exemplo a seguir. Substitua o número da conta e pelo seu próprio e pelo nome do perfil.

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

Configurando perfis AWS Glue de uso

Uma das principais vantagens de usar uma plataforma em nuvem é sua flexibilidade. No entanto, com essa facilidade de criação de recursos computacionais, surge o risco de aumentar os custos da nuvem quando não são gerenciados e sem barreiras. Como resultado, os administradores precisam evitar altos custos de infraestrutura e, ao mesmo tempo, permitir que os usuários trabalhem sem atritos desnecessários.

Com os perfis de AWS Glue uso, os administradores podem criar perfis diferentes para várias classes de usuários na conta, como desenvolvedores, testadores e equipes de produtos. Cada perfil é um conjunto exclusivo de parâmetros que podem ser atribuídos a diferentes tipos de usuários. Por exemplo, os desenvolvedores podem precisar de mais trabalhadores e ter um número maior de trabalhadores no máximo, enquanto as equipes de produto podem precisar de menos trabalhadores e de um valor menor de tempo limite ou tempo limite de inatividade.

Exemplo de comportamento de trabalhos e execuções de trabalhos

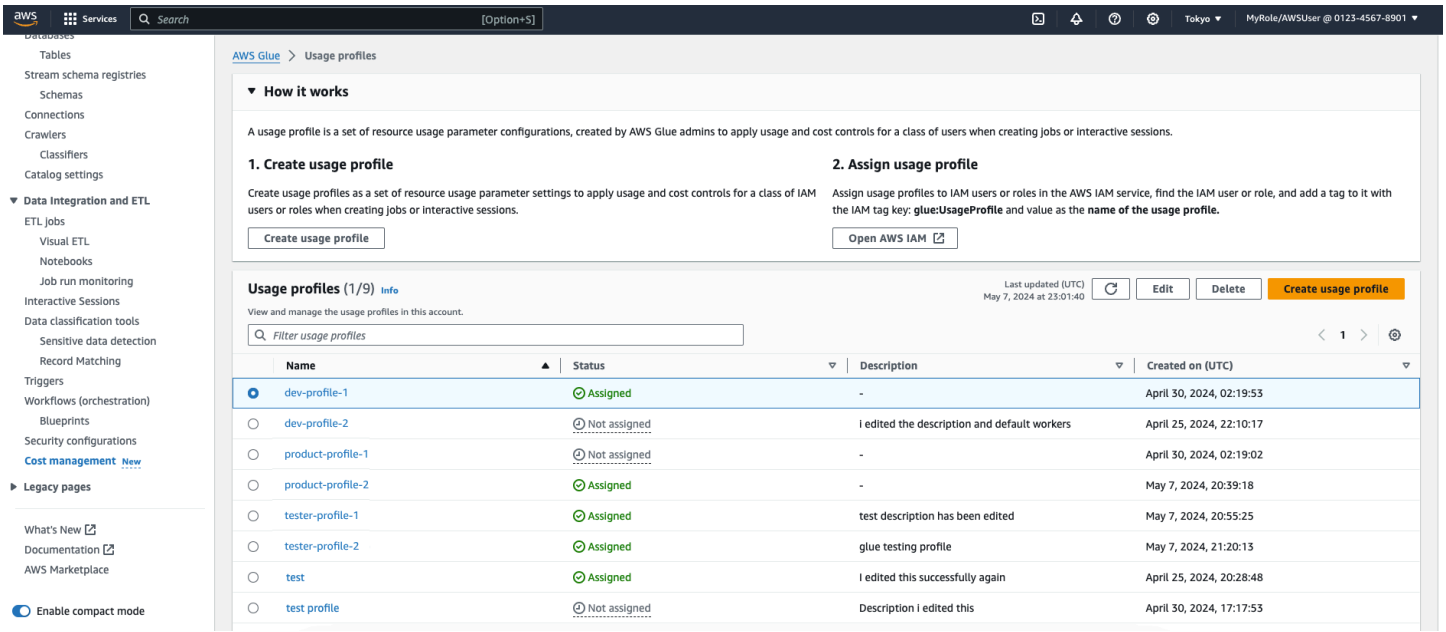
Suponha que um trabalho seja criado pelo usuário A com o perfil A. O trabalho é salvo com determinados valores de parâmetros. O usuário B com perfil B tentará executar o trabalho.

Quando o usuário A criou o trabalho, se ele não definiu um número específico de trabalhadores, o conjunto padrão no perfil do usuário A foi aplicado e salvo com as definições do trabalho.

Quando o usuário B executa o trabalho, ele é executado com os valores que foram salvos para ele. Se o perfil do usuário B for mais restritivo e não puder ser executado com tantos trabalhadores, a execução do trabalho falhará.

Perfil de uso como recurso

Um perfil de AWS Glue uso é um recurso identificado por um Amazon Resource Name (ARN). Todos os controles padrão do IAM (Identity and Access Management) se aplicam, incluindo autorização baseada em ações e recursos. Os administradores devem atualizar a política do IAM dos usuários que criam AWS Glue recursos, concedendo-lhes acesso para usar os perfis.



Tópicos

- Criação e gerenciamento de perfis de uso
- Perfis de uso e empregos

Criação e gerenciamento de perfis de uso

Criação de um perfil AWS Glue de uso

Os administradores devem criar perfis de uso e depois atribuí-los aos vários usuários. Ao criar um perfil de uso, você especifica valores padrão, bem como uma faixa de valores permitidos para vários

parâmetros de trabalho e sessão. Você deve configurar pelo menos um parâmetro para trabalhos ou sessões interativas. Você pode personalizar o valor padrão a ser usado quando um valor de parâmetro não for fornecido para o trabalho e/ou configurar um limite de intervalo ou um conjunto de valores permitidos para validação se um usuário fornecer um valor de parâmetro ao usar esse perfil.

Os padrões são uma prática recomendada definida pelo administrador para auxiliar os autores do trabalho. Quando um usuário cria um novo trabalho e não define um valor de tempo limite, o tempo limite padrão do perfil de uso será aplicado. Se o autor não tiver um perfil, os padrões AWS Glue de serviço serão aplicados e salvos na definição do trabalho. Em tempo de execução, AWS Glue impõe os limites definidos no perfil (mínimo, máximo, trabalhadores permitidos).

Depois que um parâmetro é configurado, todos os outros parâmetros são opcionais. Os parâmetros que podem ser personalizados para trabalhos ou sessões interativas são:

- Número de trabalhadores — restrinja o número de trabalhadores para evitar o uso excessivo dos recursos computacionais. Você pode definir um valor padrão, mínimo e máximo. O mínimo é 1.
- Tipo de trabalhador — restrinja os tipos de trabalhadores relevantes para suas cargas de trabalho. Você pode definir um tipo padrão e permitir tipos de trabalhadores para um perfil de usuário.
- Tempo limite — defina o tempo máximo que uma tarefa ou sessão interativa pode ser executada e consumir recursos antes de ser encerrada. Configure valores de tempo limite para evitar trabalhos de longa duração.

  Você pode definir um valor padrão, mínimo e máximo em minutos. O mínimo é 1 (minuto). Embora o tempo limite AWS Glue padrão seja de 2880 minutos, você pode definir qualquer valor padrão no perfil de uso.

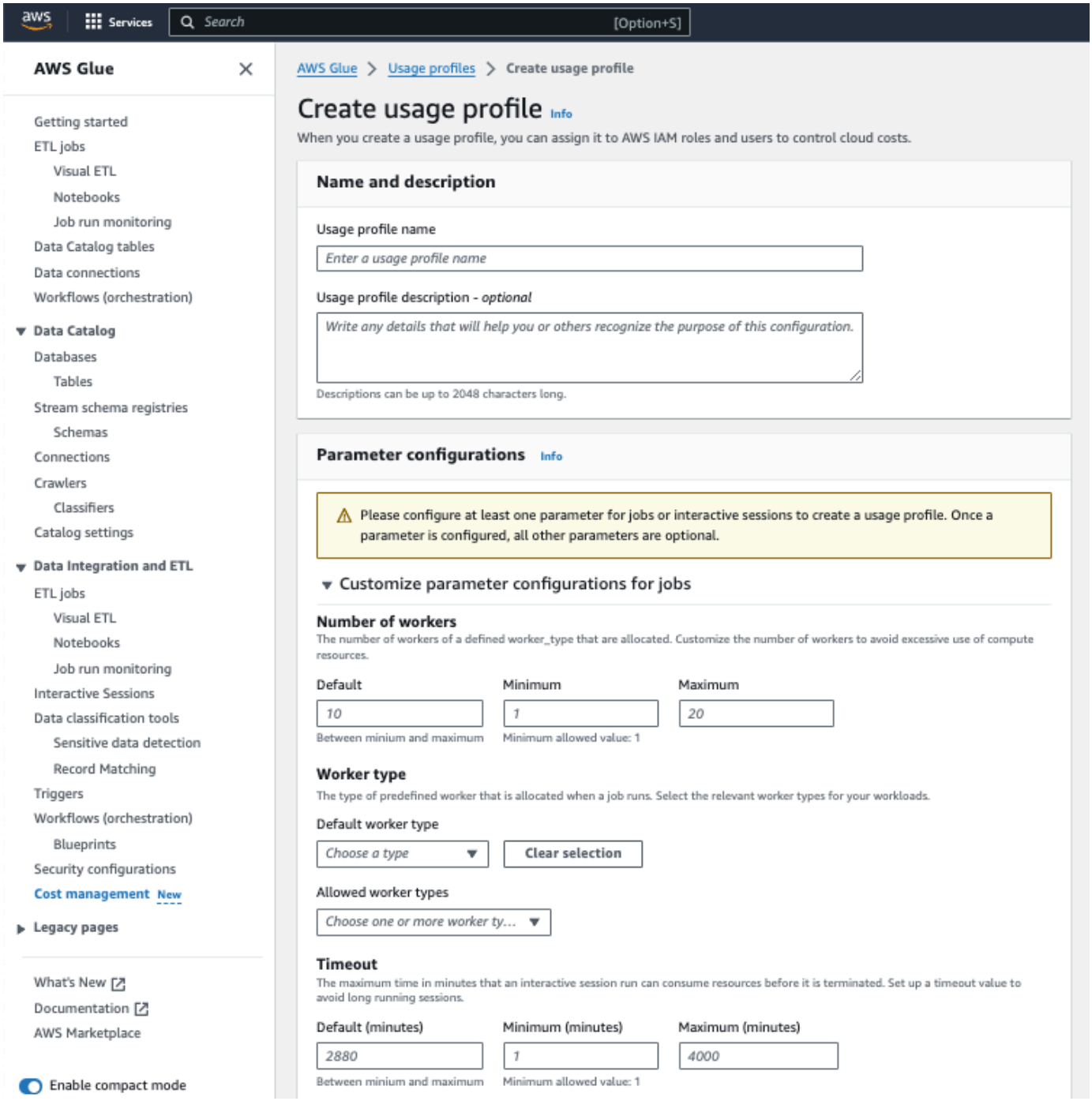
  É uma prática recomendada definir um valor para 'padrão'. Esse valor será usado para a criação do trabalho ou da sessão se nenhum valor tiver sido definido pelo usuário.
- Tempo limite de inatividade — defina o número de minutos em que uma sessão interativa fica inativa antes de expirar após a execução de uma célula. Defina o tempo limite de inatividade para que as sessões interativas sejam encerradas após a conclusão do trabalho. O intervalo de tempo limite de inatividade deve estar dentro do limite de tempo limite.

  Você pode definir um valor padrão, mínimo e máximo em minutos. O mínimo é 1 (minuto). Embora o tempo limite AWS Glue padrão seja de 2880 minutos, você pode definir qualquer valor padrão no perfil de uso.

É uma prática recomendada definir um valor para 'padrão'. Esse valor será usado para a criação da sessão se nenhum valor tiver sido definido pelo usuário.

Para criar um perfil de AWS Glue uso como administrador (console)

1. No menu de navegação à esquerda, escolha Gerenciamento de custos.
2. Escolha Criar perfil de uso.
3. Insira o nome do perfil de uso para o perfil de uso.
4. Insira uma descrição opcional que ajudará outras pessoas a reconhecer a finalidade do perfil de uso.
5. Defina pelo menos um parâmetro no perfil. Qualquer campo no formulário é um parâmetro. Por exemplo, o tempo limite mínimo de inatividade da sessão.
6. Defina todas as tags opcionais que se aplicam ao perfil de uso.
7. Escolha Salvar.



Para criar um perfil de uso (AWS CLI)

1. Insira o comando da a seguir.

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

onde o config.json pode definir valores de parâmetros para sessões interativas (`SessionConfiguration`) e jobs (): `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. Digite o comando a seguir para ver o perfil de uso criado:

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

A resposta:

```
{
    "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 adicionais usados para gerenciar perfis de uso:

- cola de cera list-usage-profiles
- *aws glue update-usage-profile --name profile-name --arquivo de configuração: //config.json*
- *aws glue delete-usage-profile --name nome do perfil*

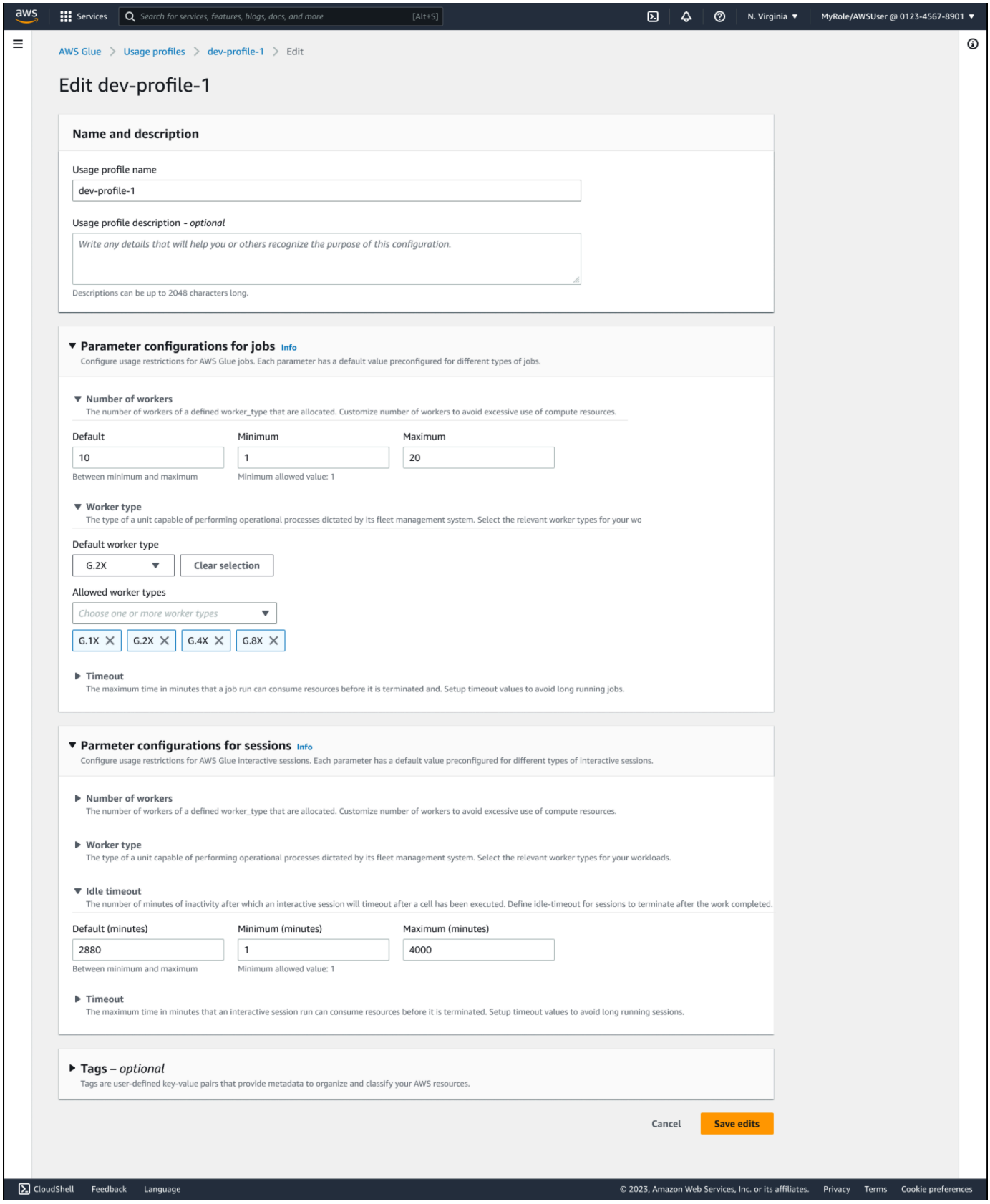
Editando um perfil de uso

Os administradores podem editar os perfis de uso que eles criaram para alterar os valores dos parâmetros do perfil para trabalhos e sessões interativas.

Para editar um perfil de uso:

Para editar um perfil de AWS Glue uso como administrador (console)

1. No menu de navegação à esquerda, escolha Gerenciamento de custos.
2. Escolha um perfil de uso que você tenha permissões para editar e escolha Editar.
3. Faça as alterações necessárias no perfil. Por padrão, os parâmetros que já têm valores são expandidos.
4. Escolha Salvar edições.


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 um perfil de uso (AWS CLI)

- Insira o comando da a seguir. A mesma sintaxe de `--configuration` arquivo é usada conforme mostrado acima no comando create.

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

onde o config.json define valores de parâmetros para sessões interativas (`SessionConfiguration`) e jobs (): `JobConfiguration`

Atribuição de um perfil de uso

A coluna Status de utilização na página Perfis de uso mostra se um perfil de uso está atribuído aos usuários. Passar o mouse sobre o status mostra as entidades do IAM atribuídas.

O administrador pode atribuir um perfil AWS Glue de uso aos usuários/funções que criam AWS Glue recursos. A atribuição de um perfil é uma combinação de duas ações:

- Atualizar a tag de usuário/função do IAM com a `glue:UsageProfile` chave e, em seguida,
- Atualização da política do IAM do usuário/função.

Para usuários que usam o AWS Glue Studio para criar trabalhos/sessões interativas, o administrador marca as seguintes funções:

- Para restrições de trabalhos, o administrador marca a função de console conectada
- Para restrições em sessões interativas, o administrador marca a função que o usuário fornece ao criar o notebook

Veja a seguir um exemplo de política que o administrador precisa atualizar sobre os usuários/funções do IAM que criam AWS Glue recursos:

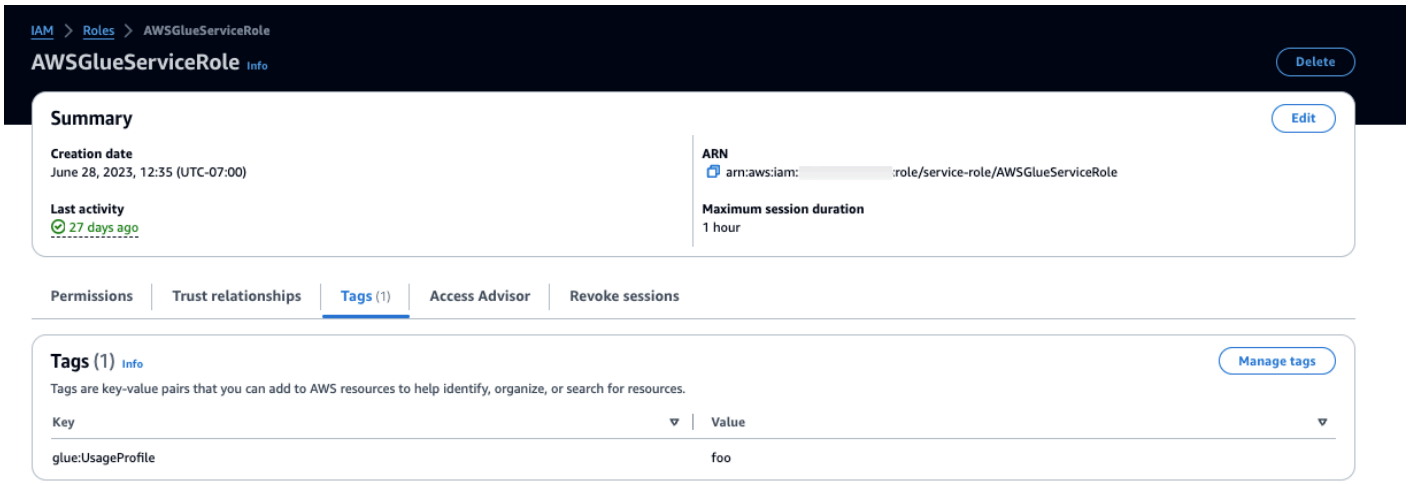
```
{
    "Effect": "Allow",
    "Action": [
        "glue:GetUsageProfile"
    ],
    "Resource": [
```

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

AWS Glue valida solicitações de trabalho, execução de tarefas e sessões com base nos valores especificados no perfil de AWS Glue uso e gera uma exceção se a solicitação não for permitida. Para APIs síncronas, um erro será gerado para o usuário. Para caminhos assíncronos, uma execução de trabalho com falha é criada com a mensagem de erro de que o parâmetro de entrada está fora do intervalo permitido para o perfil atribuído do usuário/função.

Para atribuir um perfil de uso a um usuário/função:

1. Abra o console do IAM (Identity and Access Management).
2. No painel de navegação à esquerda, escolha Usuários ou Funções.
3. Escolha um usuário ou uma função.
4. Escolha a guia Tags.
5. Selecione Adicionar nova tag
6. Adicione uma tag com a chave de `glue:UsageProfile` e o valor do nome do seu perfil de uso.
7. Selecione Save changes (Salvar alterações)



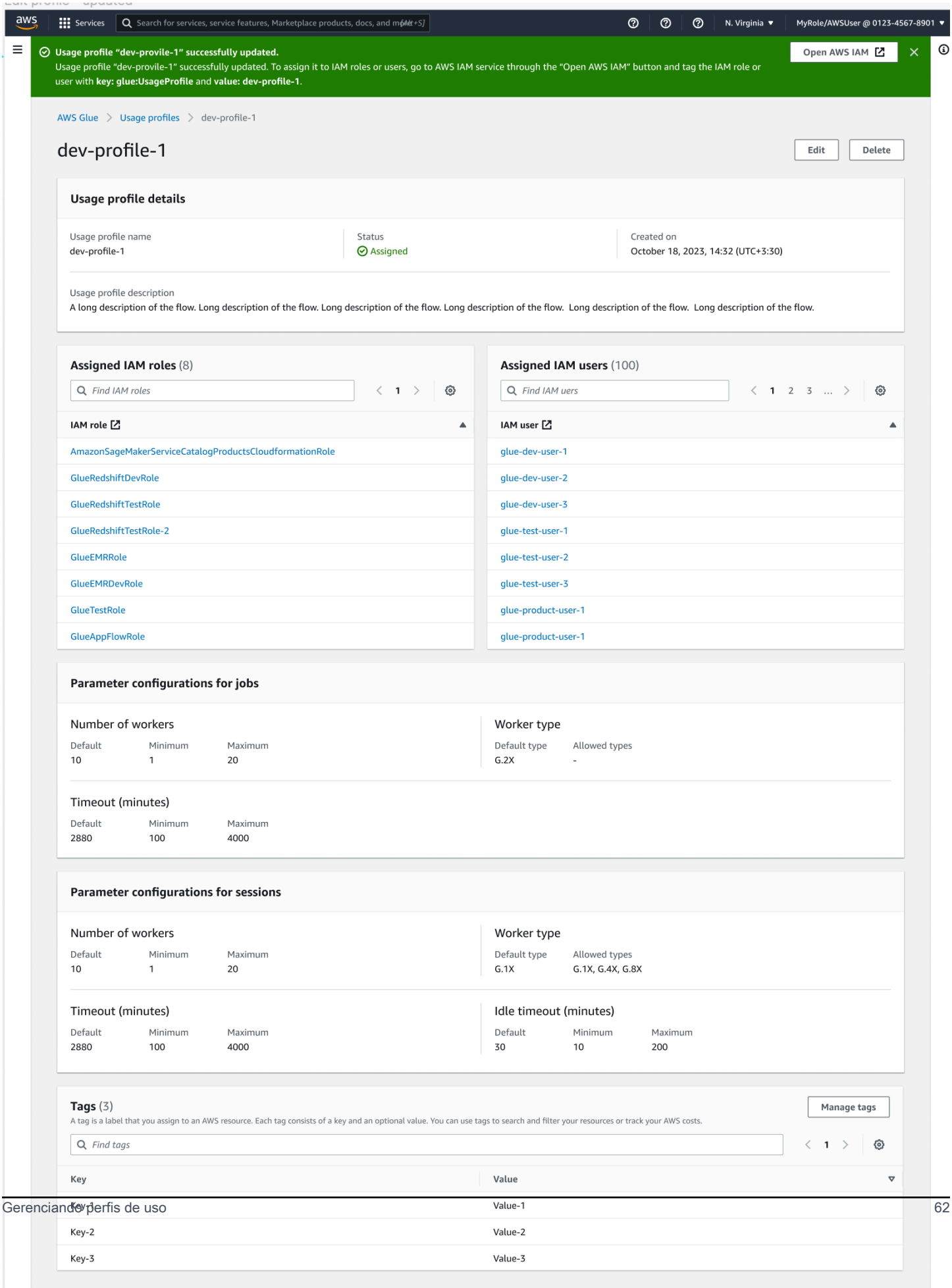
Visualizando seu perfil de uso atribuído

Os usuários podem visualizar seus perfis de uso atribuídos e usá-los ao fazer chamadas de API para criar recursos de AWS Glue trabalho e sessão ou ao iniciar um trabalho.

As permissões de perfil são fornecidas nas políticas do IAM. Desde que a política de chamadas tenha `glue:UsageProfile` permissão, o usuário poderá ver o perfil. Caso contrário, você receberá um erro de acesso negado.

Para ver um perfil de uso atribuído:

1. No menu de navegação à esquerda, escolha Gerenciamento de custos.
2. Escolha um perfil de uso que você tenha permissão para visualizar.


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

Perfis de uso e empregos

Criação de trabalhos com perfis de uso

Ao criar trabalhos, os limites e padrões definidos em seu perfil de uso serão aplicados. Seu perfil será atribuído ao trabalho ao ser salvo.

Executando trabalhos com perfis de uso

Quando você inicia uma execução de trabalho, AWS Glue impõe os limites definidos no perfil do chamador. Se não houver um chamador direto, o Glue aplicará os limites do perfil atribuído ao trabalho pelo autor.

> **Note**
>
> Quando um trabalho é executado em um cronograma (por AWS Glue fluxos de trabalho ou AWS Glue acionadores), o perfil atribuído ao trabalho que o autor aplicará.
> Quando um trabalho é executado por um serviço externo (Step Functions, MWAA) ou por uma `StartJobRun` API, o limite do perfil do chamador será imposto.

Para AWS Glue fluxos de trabalho ou AWS Glue acionadores: trabalhos preexistentes precisam ser atualizados para salvar o novo nome do perfil, de forma que os limites do perfil (mínimo, máximo e trabalhadores permitidos) sejam aplicados em tempo de execução para execuções programadas.

Visualizando um perfil de uso atribuído para trabalhos

Para ver o perfil atribuído aos seus trabalhos (que será usado em tempo de execução com AWS Glue fluxos de trabalho ou AWS Glue gatilhos programados), você pode consultar a guia Detalhes do trabalho. Você também pode ver o perfil usado em execuções anteriores na guia de detalhes de execuções de tarefas.

Atualizar ou excluir um perfil de uso anexado a um trabalho

O perfil atribuído a um trabalho é alterado após a atualização. Se o autor não tiver um perfil de uso atribuído, qualquer perfil anteriormente anexado ao trabalho será removido dele.

Conceitos básicos da AWS Glue Data Catalog

O AWS Glue Data Catalog é seu armazenamento persistente de metadados técnicos. É um serviço gerenciado que você pode usar para armazenar, anotar e compartilhar metadados na Nuvem AWS. Para ter mais informações, consulte AWS Glue Data Catalog.

O console do AWS Glue e algumas interfaces do usuário foram atualizadas recentemente.

Visão geral

É possível usar este tutorial para criar o primeiro Catálogo de dados AWS Glue, que usa um bucket do Amazon S3 como origem de dados.

Neste tutorial, você fará o seguinte usando o console do AWS Glue:

1. Criar um banco de dados.
2. Criar uma tabela
3. Usar um bucket do Amazon S3 como fonte de dados

Após concluir essas etapas, você terá usado com êxito um bucket do Amazon S3 como fonte de dados para preencher o Catálogo de dado AWS Glue.

Etapa 1: criar um banco de dados

Para começar, faça login no AWS Management Console e abra o console do AWS Glue.

Para criar um banco de dados usando o console do AWS Glue:

1. No console do AWS Glue, escolha Databases (Bancos de dados) em Data catalog (Catálogo de dados) no menu à esquerda.
2. Selecione Add database (Adicionar banco de dados).
3. Na página Criar um banco de dados, insira um nome para o banco de dados. Na seção Localização - opcional, defina a localização do URI para uso pelos clientes do catálogo de dados. Se não souber, você poderá continuar com a criação do banco de dados.
4. (Opcional). Insira uma descrição para o banco de dados.
5. Selecione Criar banco de dados.

Parabéns, você acabou de configurar seu primeiro banco de dados usando o console AWS Glue. Seu novo banco de dados aparecerá na lista de bancos de dados disponíveis. Você pode editar o banco de dados escolhendo o nome do banco de dados no painel Bancos de dados.

Próximas etapas

Outras formas de criar um banco de dados:

Você acabou de criar um banco de dados usando o console AWS Glue, mas existem outras maneiras de criar um banco de dados:

- Você pode usar crawlers para criar um banco de dados e tabelas para você automaticamente. Para configurar um banco de dados usando crawlers, consulteTrabalhar com crawlers no console AWS Glue.
- Você pode usar os modelos AWS CloudFormation. Consulte Criar recursos AWS Glue usando modelos AWS Glue Data Catalog.
- Você também pode criar um banco de dados usando as AWS Glueoperações de API do banco de dados.

  Para criar um banco de dados usando a operação `create`, estruture a solicitação, incluindo os parâmetros `DatabaseInput` (obrigatórios).

  Por exemplo:

  Veja a seguir exemplos de como você pode usar a CLI, Boto3 ou DDL para definir uma tabela com base no mesmo arquivo flight_data.csv do bucket do S3 usado no tutorial.

  CLI

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

  Boto3

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

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

```
)
```

Para obter mais informações sobre os tipos de dados, estrutura e operações da API do banco de dados, consulte API do banco de dados.

Próximas etapas

Na próxima seção, você criará uma tabela e adicionará essa tabela ao banco de dados.

Você também pode explorar as configurações e permissões do seu Catálogo de dados. Consulte Trabalhar com configurações de catálogo de dados no console AWS Glue.

Etapa 2. Criar uma tabela

Nesta etapa, você cria uma tabela usando o console AWS Glue.

1. No console AWS Glue, escolha Tables (Tabelas) no menu à esquerda.
2. Escolha Add table (Adicionar tabela).
3. Defina as propriedades da tabela inserindo um nome para a tabela em Table details (Detalhes da tabela).
4. No seção Databases (Banco de dados), escolha no menu suspenso o banco de dados que criou na Etapa 1.
5. Na seção Add a data store (Adicionar um datastore), a opção S3 será selecionada por padrão como o tipo de fonte.
6. Em Data is located in (Dados localizados em), escolha Specified path in another account (Caminho especificado em outra conta).
7. Copie e cole o caminho para o campo de entrada Include path (Incluir caminho):

   ```
   s3://crawler-public-us-west-2/flight/2016/csv/
   ```
8. Na seção Data format (Formato de dados), para Classification (Classificação), escolha CSV e para Delimiter (Delimitador), escolha comma (,) (vírgula [,]). Escolha Próximo.
9. Será solicitado que você defina um esquema. O esquema define a estrutura e o formato de um registro de dados. Selecione Add column (Adicionar coluna). (Para obter mais informações, consulte Schema registries (Registros de esquema).
10. Especifique as propriedades da coluna:

a. Insira um nome de coluna.
b. Para o Column type (Tipo de coluna), 'string' já está selecionada por padrão.
c. Para o Column number (Número da coluna), 'string' já está selecionada por padrão.
d. Escolha Add.

11. Você é solicitado a adicionar índices de partição. Isso é opcional. Para pular esta etapa, escolha, escolha Next (Próximo).
12. Um resumo das propriedades da tabela é exibido. Se tudo estiver conforme o esperado, escolha Criar. Caso contrário, escolha Voltar e faça edições conforme for necessário.

Parabéns, você criou uma tabela manualmente e a associou a um banco de dados. Sua tabela recém-criada aparecerá no painel Tabelas. No painel, você pode modificar e gerenciar suas tabelas.

Para obter mais informações, consulte Working with Tables (Trabalhar com tabelas) no console AWS Glue.

Next steps (Próximas etapas)

Next steps (Próximas etapas)

Agora que o Catálogo de dados está preenchido, você pode começar a criar trabalhos no AWS Glue. Consulte Criar trabalhos de ETL visual com o AWS Glue Studio.

Além de usar o console, há outras maneiras de definir tabelas no Catálogo de dados, incluindo:

- Criar e executar um crawler
- Adicionar classificadores a um crawler no AWS Glue
- Usar a API da tabelaAWS Glue
- Usar o modelo AWS Glue Data Catalog
- Migrar um metastore do Apache Hive
- Usar o AWS CLI, Boto3 ou linguagem de definição de dados (DDL)

  Veja a seguir exemplos de como você pode usar a CLI, Boto3 ou DDL para definir uma tabela com base no mesmo arquivo flight_data.csv do bucket do S3 usado no tutorial.

  Consulte a documentação sobre como estruturar um comando AWS CLI. O exemplo de CLI contém a sintaxe JSON para o 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')
```

Configurar o acesso de rede aos armazenamentos de dados

Para executar trabalhos de extração, transformação e carregamento (ETL), o AWS Glue precisa de acesso aos armazenamentos de dados. Se não for necessário executar um trabalho na sub-rede da sua nuvem privada virtual (VPC), por exemplo: transformação de dados do Amazon S3 para o Amazon S3, nenhuma configuração adicional será necessária.

Se for necessário executar um trabalho na sub-rede de sua VPC, por exemplo, transformar dados de um armazenamento de dados JDBC em uma sub-rede privada, o AWS Glue configurará interfaces de rede elástica que permitem que os trabalhos se conectem com segurança a outros recursos dentro da sua VPC. Cada interface de rede elástica recebe um endereço IP privado do intervalo de endereços IP na sub-rede que você especificou. Nenhum endereço IP público atribuído. Grupos de segurança especificados na conexão do AWS Glue são aplicados a cada uma das interfaces de rede elásticas. Para ter mais informações, consulte Configurar um Amazon VPC para conexões JDBC aos armazenamentos de dados do Amazon RDS desde o AWS Glue.

Todos os armazenamentos de dados JDBC acessados pelo trabalho devem estar disponíveis na sub-rede da VPC. Para acessar o Amazon S3 na sua VPC, é necessário ter um endpoint da VPC. Se o seu trabalho precisar acessar recursos da VPC e a Internet pública, a VPC precisará conter uma instância de gateway NAT (Network Address Translation).

Um trabalho ou endpoint de desenvolvimento pode acessar somente uma VPC (e sub-rede) por vez. Se você precisar acessar armazenamentos de dados em várias VPCs, as opções são as seguintes:

- Use o emparelhamento de VPCs para acessar os armazenamentos de dados. Para obter mais informações sobre o emparelhamento de VPCs, consulte Conceitos básicos sobre o emparelhamento de VPCs
- Use um bucket do Amazon S3 como um local de armazenamento intermediário. Divida o trabalho em dois, com a saída do Amazon S3 do trabalho 1 como a entrada do trabalho 2.

Para obter detalhes sobre como conectar a um armazenamento de dados do Amazon Redshift usando o Amazon VPC, consulte the section called “Configurar o Redshift”.

Para obter detalhes sobre como conectar a um armazenamento de dados do Amazon RDS usando o Amazon VPC, consulte the section called “Configurar um Amazon VPC para conectar aos armazenamentos de dados do Amazon RDS”.

Depois que as regras necessárias são definidas no Amazon VPC, você cria uma conexão no AWS Glue com as propriedades necessárias para conectar com seus armazenamentos de dados. Para obter mais informações sobre a conexão, consulte Conectar a dados.

> **Note**
>
> Configure seu ambiente de DNS para o AWS Glue. Para ter mais informações, consulte Configurar o DNS na VPC.

Tópicos

Configurar uma VPC para se conectar ao PyPI para AWS Glue

O Python Package Index (PyPI) é um repositório de software para a linguagem de programação Python. Este tópico aborda os detalhes necessários para oferecer suporte ao uso de pacotes instalados pelo pip (conforme especificado pelo criador da sessão usando o sinalizador `--additional-python-modules`).

O uso de sessões interativas do AWS Glue com um conector resulta no uso da rede VPC por meio da sub-rede especificada para o conector. Consequentemente, os serviços do AWS e outros destinos de rede não estão disponíveis, a menos que você defina uma configuração especial.

As resoluções para esse problema incluem:

- Uso de um gateway da internet que seja acessível por sua sessão.
- Configuração e uso de um bucket S3 com um repositório PyPI/simple contendo o fechamento transitivo das dependências de um conjunto de pacotes.
- Uso de um repositório CodeArtifact que está espelhando o PyPI e anexado à sua VPC.

Configuração de um gateway da Internet

Os aspectos técnicos são detalhados em Casos de uso do gateway NAT, mas observe esses requisitos para usar o `--additional-python-modules`. Especificamente, o `--additional-`

`python-modules` requer acesso ao pypi.org, que é determinado pela configuração da sua VPC. Observe os seguintes requisitos:

1. A exigência de instalar módulos adicionais do python via pip install para uma sessão do usuário. Se a sessão usar um conector, sua configuração poderá ser afetada.
2. Quando um conector está sendo usado com o `--additional-python-modules`, quando a sessão é iniciada, a sub-rede associada aos `PhysicalConnectionRequirements` do conector deve fornecer um caminho de rede para alcançar pypi.org.
3. Você deve determinar se sua configuração está correta ou não.

Configurar um bucket do Amazon S3 para hospedar um repositório PyPI/simple de destino

Este exemplo configura um espelho PyPI no Amazon S3 para um conjunto de pacotes e suas dependências.

Para configurar o espelho PyPI para um conjunto de pacotes:

```
# 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/*
```

Se você já tiver um repositório de artefatos existente, ele terá um URL de índice para uso do pip que você pode fornecer no lugar do URL de exemplo para o bucket do Amazon S3, conforme descrito acima.

Para usar o index-url personalizado, com alguns pacotes de exemplo:

```
%%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"
}
```

Configurar um espelho CodeArtifact de pypi conectado à sua VPC

Para configurar um espelho:

1. Crie um repositório na mesma região da sub-rede usada pelo conector.

   Selecione `Public upstream repositories` e escolha `pypi-store`.
2. Forneça acesso ao repositório da VPC para a sub-rede.
3. Especifique o `--index-url` correto usando o `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 obter mais informações, consulte Use CodeArtifact from a VPC.

Configurar o DNS na VPC

Domain Name System (DNS) é um padrão por meio do qual os nomes usados na Internet são determinados de acordo com os endereços IP correspondentes. Um nome de host do DNS denomina exclusivamente um computador e consiste em um nome de host e em um nome de domínio. Os servidores DNS determinam os nomes do host DNS de acordo com os endereços IP correspondentes.

Para configurar o DNS na sua VPC, certifique-se de que os nomes de host do DNS e a resolução de DNS estão habilitados na VPC. Os atributos de rede `enableDnsHostnames` e `enableDnsSupport` da VPC precisam ser definidos como `true`. Para visualizar e modificar esses atributos, acesse o console da VPC em https://console.aws.amazon.com/vpc/.

Para obter mais informações, consulte Como usar o DNS com sua VPC. Além disso, é possível usar a AWS CLI e chamar o comando modify-vpc-attribute para configurar os atributos de rede da VPC.

> **Note**
>
> Se você estiver usando o Route 53, confirme se a sua configuração não substitui os atributos de rede do DNS.

Configurar criptografia no AWS Glue

O seguinte exemplo de fluxo de trabalho destaca as opções a serem configuradas quando você usa criptografia com AWS Glue. O exemplo demonstra o uso de chaves específicas do AWS Key Management Service (AWS KMS), mas você pode escolher outras configurações com base nas suas necessidades específicas. Esse fluxo de trabalho destaca apenas as opções que pertencem a criptografia ao configurar o AWS Glue.

1. Se o usuário do console do AWS Glue não usar uma política de permissões que permita todas as operações de API do AWS Glue (por exemplo, `"glue:*"`), verifique se as seguintes ações são permitidas:
   - `"glue:GetDataCatalogEncryptionSettings"`
   - `"glue:PutDataCatalogEncryptionSettings"`
   - `"glue:CreateSecurityConfiguration"`
   - `"glue:GetSecurityConfiguration"`
   - `"glue:GetSecurityConfigurations"`
   - `"glue:DeleteSecurityConfiguration"`
2. Qualquer cliente que acesse ou grave em um catálogo criptografado, ou seja, qualquer usuário do console, crawler, trabalho ou endpoint de desenvolvimento, precisa das permissões a seguir.

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

3. Qualquer usuário ou função que acesse uma senha de conexão criptografada precisa das permissões a seguir.

```
{
 "Version": "2012-10-17",
```

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

4. A função de qualquer trabalho de extração, transformação e carregamento (ETL) que grave dados criptografados no Amazon S3 precisa das permissões a seguir.

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

5. Qualquer crawler ou trabalho de ETL que grave Amazon CloudWatch Logs criptografados requer as permissões a seguir nas políticas de chave e do IAM.

   Na política de chave (não na política do 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 obter mais informações sobre políticas de chave, consulte Usar políticas de chave no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

Na política do IAM, anexe a permissão `logs:AssociateKmsKey`:

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

6. Qualquer trabalho de ETL que use um marcador de trabalho criptografado precisa das permissões a seguir.

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

7. No console do AWS Glue, selecione Settings (Configurações) no painel de navegação.
   a. Na página Data catalog settings (Configurações do catálogo de dados), criptografe o Data Catalog selecionando Metadata encryption (Criptografia de metadados). Essa opção criptografa todos os objetos no Data Catalog com a chave do AWS KMS que você escolher.
   b. Em Chave do AWS KMS, escolha aws/glue. Também é possível escolher uma chave do AWS KMS criada por você.

> **Important**
>
> O AWS Glue só oferece suporte a chaves mestras do cliente (CMKs) simétricas. A lista de chaves do AWS KMS exibe apenas chaves simétricas. No entanto, se você selecionar Escolher um ARN de chave AWS KMS, o console permitirá que você insira um ARN para qualquer tipo de chave. Certifique-se de inserir apenas ARNs para chaves simétricas.

Quando a criptografia estiver habilitada, o cliente que estiver acessando o Data Catalog deverá ter permissões do AWS KMS.

8. No painel de navegação, escolha Security configurations (Configurações de segurança). Uma configuração de segurança é um conjunto de propriedades de segurança que pode ser usado para configurar processos do AWS Glue. Em seguida, escolha Add security configuration (Adicionar configuração de segurança). Na configuração, escolha qualquer uma das opções a seguir:

   a. Selecione S3 encryption (Criptografia do S3). Para Encryption mode (Modo de criptografia), escolha SSE-KMS. Para a chave do AWS KMS, escolha aws/s3 (certifique-se de que o usuário tenha permissão para usar essa chave). Isso permite que os dados gravados pelo trabalho no Amazon S3 usem a chave do AWS KMS do AWS Glue gerenciada pela AWS.

   b. Selecione CloudWatch logs encryption (Criptografia do CloudWatch Logs) e escolha uma CMK. (Verifique se o usuário tem permissão para usar essa chave). Para obter mais informações, consulte Criptografar dados de log no CloudWatch Logs usando o AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

      > **Important**
      >
      > O AWS Glue só oferece suporte a chaves mestras do cliente (CMKs) simétricas. A lista de chaves do AWS KMS exibe apenas chaves simétricas. No entanto, se você selecionar Escolher um ARN de chave AWS KMS, o console permitirá que você insira um ARN para qualquer tipo de chave. Certifique-se de inserir apenas ARNs para chaves simétricas.

   c. Selecione Advanced properties (Propriedades avançadas) e Job bookmark encryption (Criptografia do marcador de trabalho). Para a chave do AWS KMS, escolha aws/glue (certifique-se de que o usuário tenha permissão para usar essa chave). Isso habilita a

criptografia de marcadores de trabalho gravados no Amazon S3 com a chave do AWS KMS do AWS Glue.

9. No painel de navegação, escolha Connections.
   a. Escolha Add connection (Adicionar conexão) para criar uma conexão com o armazenamento de dados JDBC (Java Database Connectivity) que é o destino do seu trabalho de ETL.
   b. Para garantir que a criptografia do Secure Sockets Layer (SSL) seja usada, selecione Require SSL connection (Exigir conexão SSL) e teste a conexão.
10. No painel de navegação, escolha Tarefas.
    a. Escolha Add job (Adicionar trabalho) para criar um trabalho que transforme dados.
    b. Na definição de tarefa, escolha a configuração de segurança que você criou.
11. No console do AWS Glue, execute o trabalho sob demanda. Verifique se os dados do Amazon S3 e os CloudWatch Logs gravados pelo trabalho, e os marcadores de trabalho, estão criptografados.

Configurar redes para desenvolvimento para o AWS Glue

Para executar seus scripts de extração, transformação e carregamento (ETL) com o AWS Glue, você pode desenvolvê-los e testá-los usando um endpoint de desenvolvimento. Os endpoints de desenvolvimento não são suportados para uso com trabalhos do AWS Glue versão 2.0. Para as versões 2.0 e posteriores, o método de desenvolvimento preferido é usar o Jupyter Notebook com um dos kernels AWS Glue. Para ter mais informações, consulte the section called “Conceitos básicos das sessões interativas do AWS Glue”.

Configurar a rede para um endpoint de desenvolvimento

Ao configurar um endpoint de desenvolvimento, você especifica uma nuvem virtual privada (VPC), uma sub-rede e security groups.

> **Note**
>
> Configure seu ambiente de DNS para o AWS Glue. Para ter mais informações, consulte Configurar o DNS na VPC.

Para permitir que o AWS Glue acesse os recursos necessários, adicione uma linha na tabela de rotas da sua sub-rede para associar uma lista de prefixos para o Amazon S3 ao endpoint da VPC.

É necessário um ID de lista de prefixos para criar uma regra de grupo de segurança de saída que permita que o tráfego de uma VPC acesse um produto da AWS por meio de um endpoint da VPC. Para facilitar a conexão com um servidor de cadernos associado a esse endpoint de desenvolvimento, na máquina local, adicione uma linha à tabela de rotas para incluir um ID de gateway da Internet. Para obter mais informações, consulte VPC Endpoints. Atualize a tabela de rotas da sub-rede de modo que ela fique semelhante a esta tabela:

Destination (Destino)	Destino		
10.0.0.0/16	local		
pl-id para o Amazon S3	vpce-id		
0.0.0.0/0	igw-xxxx		

Para permitir que o AWS Glue se comunique entre seus componentes, especifique um grupo de segurança com uma regra de entrada de autorreferência para todas as portas TCP. Ao criar uma regra de autorreferência, você pode restringir a origem ao mesmo security group na VPC e fechá-la para todas as redes. O security group padrão para sua VPC pode já conter uma regra de entrada de autorreferenciada para ALL Traffic.

Para configurar um security group

1. Faça login no AWS Management Console e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. No painel de navegação esquerdo, escolha Security Groups.
3. Escolha um security group existente na lista ou Create Security Group para usar com o endpoint de desenvolvimento.
4. No painel do security group, navegue até a guia Inbound.
5. Adicione uma regra de autorreferência para permitir que os componentes do AWS Glue se comuniquem. Especificamente, adicione ou confirme que existe uma regra de Type `All TCP`, que Protocol é `TCP`, Port Range inclui todas as portas e Source e Group ID apresentam o mesmo nome de security group.

   A regra de entrada é semelhante a esta:

Tipo	Protocolo	Intervalo de portas	Origem
Todos os TCP	TCP	0–65535	*security-group*

Veja a seguir um exemplo de uma regra de entrada de autorreferência:

6. Adicione uma regra para o tráfego de saída também. Abra um tráfego de saída para todas as portas ou crie uma regra de autorreferência de Type `All TCP`. Protocol deve ser `TCP`, Port Range deve incluir todas as portas e Source e Group ID devem apresentar o mesmo nome de security group.

   A regra de saída é semelhante a uma dessas regras:

Tipo	Protocolo	Intervalo de portas	Destino
Todos os TCP	TCP	0–65535	*security-group*
Todo o tráfego	ALL	ALL	0.0.0.0/0

Configurar o Amazon EC2 para um servidor de cadernos

Com um endpoint de desenvolvimento, você pode criar um servidor de caderno para testar seus scripts de ETL com cadernos Jupyter. Para ativar a comunicação com seu notebook, especifique um security group com regras de entrada para HTTPS (porta 443) e SSH (porta 22). Verifique se a origem da regra é 0.0.0.0/0 ou o endereço IP da máquina que está se conectando ao notebook.

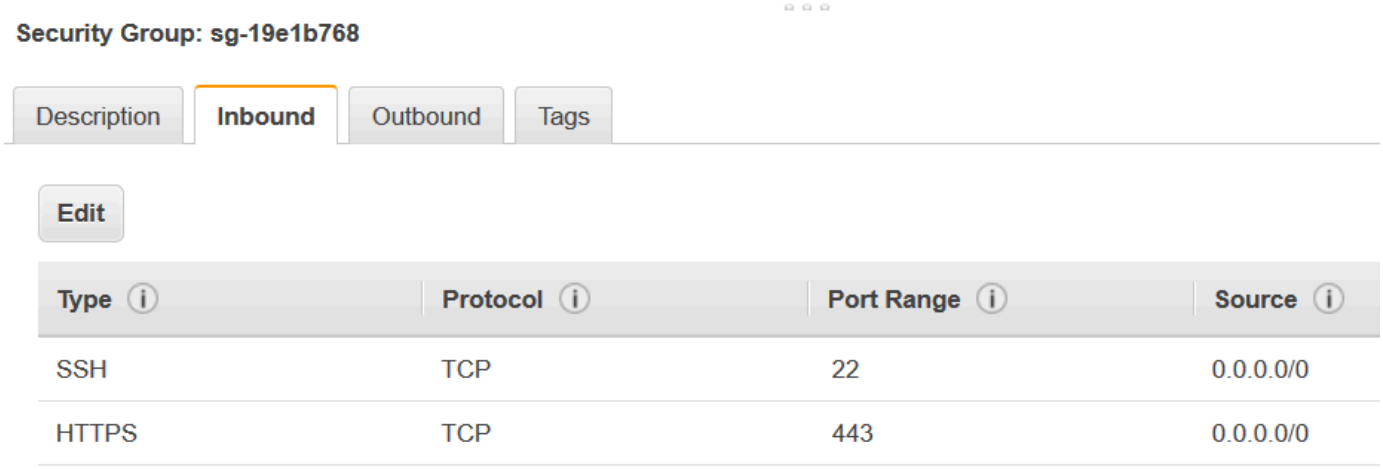
Para configurar um security group

1. Faça login no AWS Management Console e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. No painel de navegação esquerdo, escolha Security Groups.
3. Escolha um security group existente na lista ou Create Security Group para usar com o servidor de notebook. O security group associado ao seu endpoint de desenvolvimento também é usado para criar seu servidor de notebook.
4. No painel do security group, navegue até a guia Inbound.

5. Adicione regras de entrada semelhantes a esta:

Tipo	Protocolo	Intervalo de portas	Origem
SSH	TCP	22	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0

Veja a seguir um exemplo das regras de entrada para o security group:



Descoberta e catalogação de dados no AWS Glue

O AWS Glue Data Catalog é um repositório centralizado que armazena metadados sobre os conjuntos de dados da sua organização. Ele funciona como um índice para métricas de localização, esquema e runtime das suas fontes de dados. Os metadados são armazenados em tabelas de metadados, onde cada tabela representa um único armazenamento de dados.

O Catálogo de Dados pode ser preenchido com a ajuda de um crawler, que verifica automaticamente suas fontes de dados e extrai metadados. Um crawler pode se conectar a fontes de dados internas (baseadas na AWS) e externas à AWS.

Para obter mais informações sobre as fontes de dados compatíveis, consulte Quais armazenamentos de dados posso rastrear?.

Também é possível criar tabelas no Catálogo de Dados manualmente definindo a estrutura da tabela, o esquema e a estrutura de particionamento de acordo com seus requisitos específicos.

Para obter mais informações sobre como criar manualmente tabelas de bancos de dados, consulte Definir metadados manualmente.

Use as informações no Catálogo de Dados para criar e monitorar seus trabalhos de ETL. O Catálogo de Dados se integra a outros serviços de análise da AWS, fornecendo uma visão unificada das fontes de dados e facilitando o gerenciamento e a análise de dados.

- Amazon Athena: armazene e consulte metadados da tabela no Catálogo de Dados para os dados do Amazon S3 usando SQL.
- AWS Lake Formation: defina e gerencie centralmente políticas refinadas de acesso a dados e audite o acesso aos dados.
- Amazon EMR: acesse fontes de dados definidas no Catálogo de Dados para processamento de big data.
- Amazon SageMaker: crie, treine e implemente modelos de machine learning com rapidez e confiança.

Principais recursos do Catálogo de Dados

A seguir estão os principais aspectos do Catálogo de Dados.

Repositório de metadados

O Catálogo de Dados atua como um repositório central de metadados, armazenando informações sobre a localização, o esquema e as propriedades das suas fontes de dados. Esses metadados são organizados em bancos de dados e tabelas de forma semelhante a um catálogo de banco de dados relacional tradicional.

Descoberta automática de dados

Os Crawler do AWS Glues podem descobrir e catalogar automaticamente fontes de dados novas ou atualizadas, reduzindo a sobrecarga do gerenciamento manual de metadados e garantindo que seu Catálogo de Dados permaneça atualizado. Ao catalogar suas fontes de dados, o Catálogo de Dados facilita que usuários e aplicações descubram e entendam os ativos de dados disponíveis em sua organização, promovendo a reutilização e a colaboração de dados.

O catálogo de dados oferece suporte a uma ampla variedade de fontes de dados, incluindo Amazon S3, Amazon RDS, Amazon Redshift, Apache Hive e muito mais. Ele pode inferir e armazenar automaticamente metadados dessas fontes usando Crawler do AWS Glues.

Para obter mais informações, consulte Usar crawlers para preencher o catálogo de dados .

Gerenciamento de esquemas

O Catálogo de Dados captura e gerencia automaticamente o esquema de suas fontes de dados, incluindo inferência, evolução e controle de versão do esquema. É possível atualizar os esquemas e as partições no Catálogo de Dados usando trabalhos do AWS Glue ETL.

Otimização de tabelas

Para obter uma melhor performance de leitura por serviços de análise da AWS, como o Amazon Athena e o Amazon EMR e trabalhos de ETL do AWS Glue, o Catálogo de Dados oferece compactação gerenciada (um processo que compacta pequenos objetos do Amazon S3 em objetos maiores) para tabelas do Iceberg no Catálogo de Dados. Você pode usar o console do AWS Glue, o console do AWS Lake Formation, a AWS CLI ou a API da AWS para habilitar ou desabilitar a compactação de tabelas individuais do Iceberg que estão no Catálogo de Dados.

Para ter mais informações, consulte Otimizar tabelas Iceberg.

Estatísticas de colunas

Você pode calcular estatísticas em nível de coluna para tabelas do Catálogo de Dados em formatos de dados como Parquet, ORC, JSON, ION, CSV e XML sem precisar configurar

pipelines de dados adicionais. As estatísticas de colunas ajudam você a entender os perfis de dados obtendo insights sobre os valores em uma coluna. O Catálogo de Dados possibilita a geração de estatísticas para valores de colunas, como valor mínimo, valor máximo, valores nulos totais, valores distintos totais, comprimento médio dos valores e ocorrências totais de valores reais.

Para ter mais informações, consulte Otimizar a performance da consulta usando estatísticas de coluna.

Linhagem de dados

O Catálogo de Dados mantém um registro das transformações e operações realizadas em seus dados, fornecendo informações sobre a linhagem de dados. Essas informações de linhagem são valiosas para auditoria, conformidade e compreensão da proveniência dos dados.

Integração com outros serviços da AWS

O catálogo de dados se integra perfeitamente a outros serviços da AWS, como AWS Lake Formation, Amazon Athena, Amazon Redshift Spectrum e Amazon EMR. Essa integração permite que você consulte e analise dados em vários armazenamentos de dados usando uma camada de metadados única e consistente.

Segurança e controle de acesso

O AWS Glue se integra ao AWS Lake Formation para oferecer suporte ao controle de acesso refinado aos recursos do Catálogo de Dados, permitindo que você gerencie permissões e proteja o acesso aos seus ativos de dados com base nas políticas e requisitos da sua organização. O AWS Glue se integra ao AWS Key Management Service (AWS KMS) para criptografar metadados armazenados no Catálogo de Dados.

Tópicos

- Preencher o AWS Glue Data Catalog
- Preencher e gerenciar tabelas transacionais
- Gerenciar o Catálogo de Dados
- Acessar o Catálogo de Dados
- Práticas recomendadas do Catálogo de Dados do AWS Glue
- Registro de esquemas do AWS Glue

Preencher o AWS Glue Data Catalog

É possível preencher o AWS Glue Data Catalog usando os seguintes métodos:

- Crawler do AWS Glue: um Crawler do AWS Glue pode descobrir e catalogar automaticamente fontes de dados, como bancos de dados, data lakes e dados de streaming. Os crawlers são o método mais comum e recomendado para preencher o Catálogo de Dados, pois eles podem descobrir e inferir automaticamente metadados de uma ampla variedade de fontes de dados.
- Adicionar metadados manualmente: é possível definir manualmente bancos de dados, tabelas e detalhes de conexão e adicioná-los ao Catálogo de Dados usando o console do AWS Glue, o console do Lake Formation, a AWS CLI ou as APIs do AWS Glue. A entrada manual é útil quando você deseja catalogar fontes de dados que não podem ser obtidas por crawling.
- Integração com outros serviços da AWS: é possível preencher o catálogo de dados com metadados de serviços como o AWS Lake Formation e o Amazon Athena. Esses serviços podem descobrir e registrar fontes de dados no Catálogo de Dados.
- Preencher a partir de um repositório de metadados existente: se você tiver um repositório de metadados existente, como o Apache Hive Metastore, poderá usar o AWS Glue para importar esses metadados para o Catálogo de Dados. Para obter mais informações, consulte Migração entre o Hive Metastore e o AWS Glue Data Catalog no GitHub.

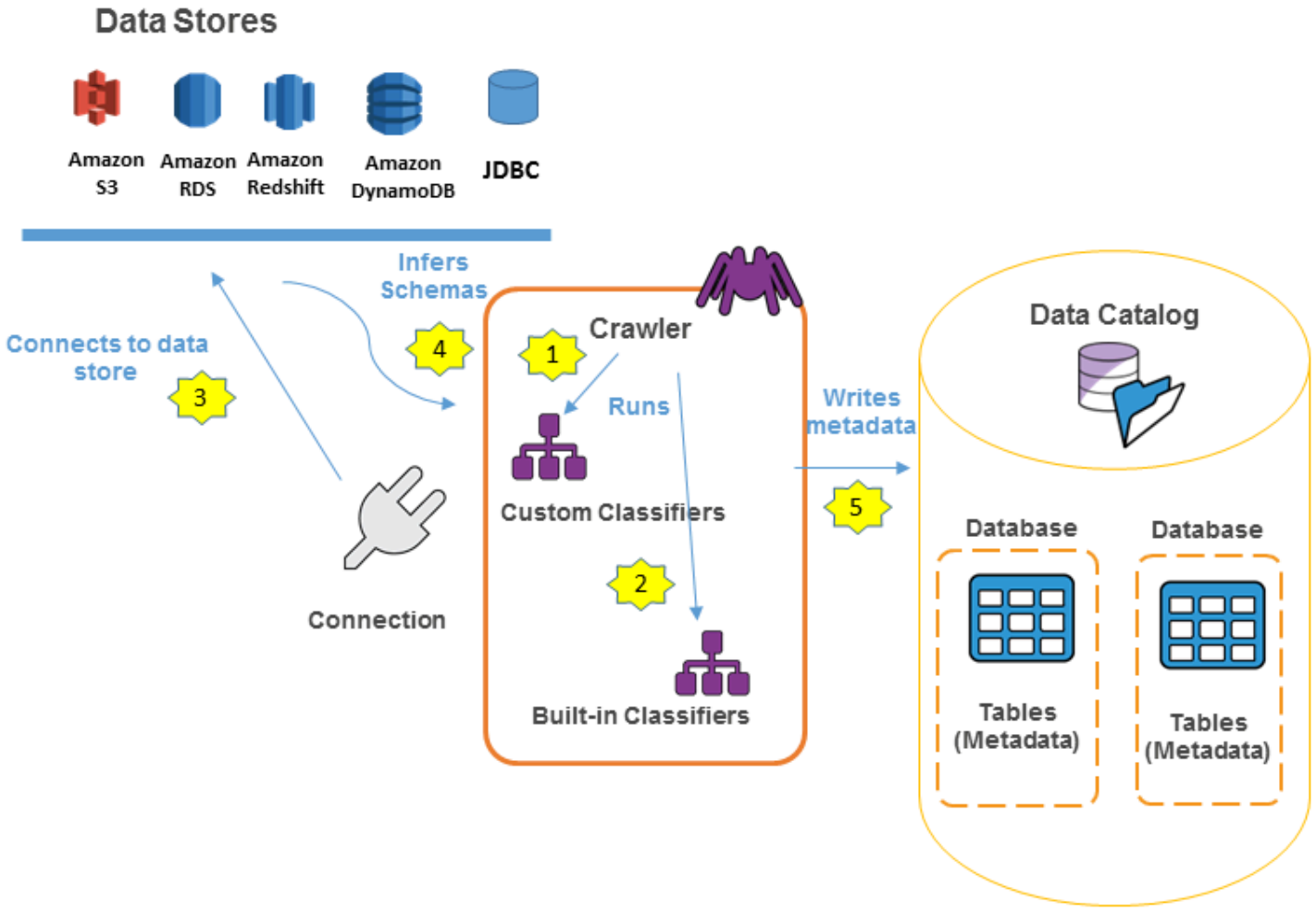
Tópicos

Usar crawlers para preencher o catálogo de dados

Você pode usar um Crawler do AWS Glue para preencher o AWS Glue Data Catalog com bancos de dados e tabelas. Este é o principal método usado pela maioria dos usuários do AWS Glue. Um crawler pode rastrear vários armazenamentos de dados em uma única execução. Após a conclusão, o crawler cria ou atualiza uma ou mais tabelas no Data Catalog. As tarefas de extração, transformação e carregamento (ETL) que você define no AWS Glue usam essas tabelas do Data Catalog como fontes e destinos. O trabalho de ETL lê e grava os armazenamentos de dados que são especificados nas tabelas do Data Catalog de fonte e de destino.

Fluxo de trabalho

O seguinte diagrama de fluxo de trabalho mostra como os crawlers do AWS Glue interagem com os armazenamentos de dados e outros elementos para preencher o Data Catalog.



Veja a seguir o fluxo de trabalho geral sobre como um crawler preenche o AWS Glue Data Catalog:

1. Um crawler executa todos os classificadores personalizados que você escolhe para inferir o formato e o esquema dos seus dados. Você fornece o código para classificadores personalizados, e eles são executados na ordem especificada.

   O primeiro classificador personalizado a reconhecer com sucesso a estrutura de dados é usado para criar um esquema. Os classificadores personalizados em posições inferiores na lista são ignorados.

2. Se nenhum classificador personalizado corresponder ao esquema dos seus dados, os classificadores integrados tentarão reconhecê-lo. Um exemplo de um classificador integrado é um que reconhece JSON.
3. O crawler se conecta ao armazenamento de dados. Alguns armazenamentos de dados requerem propriedades de conexão para o acesso ao crawler.
4. O esquema inferido é criado para os seus dados.
5. O crawler grava os metadados no Data Catalog. Uma definição de tabela contém metadados sobre os dados no seu armazenamento de dados. A tabela é gravada em um banco de dados, que é um contêiner de tabelas no Data Catalog. Os atributos de uma tabela incluem a classificação, que é um rótulo criado pelo classificador que inferiu o esquema da tabela.

Tópicos

Como funcionam os crawlers

Quando um crawler é executado, ele obtém as ações a seguir para interrogar um armazenamento de dados:

- Classifica dados para determinar o formato, o esquema e as propriedades associadas de dados brutos – Você pode configurar os resultados de classificação criando um classificador personalizado.
- Agrupa dados em tabelas ou partições – Os dados são agrupados com base na heurística do crawler.

- Grava metadados no Data Catalog: você pode configurar como o crawler adiciona, atualiza e exclui tabelas e partições.

Ao definir um crawler, você escolhe um ou mais classificadores que avaliam o formato dos seus dados para inferir um esquema. Quando o crawler é executado, o primeiro classificador da sua lista a reconhecer com sucesso seu armazenamento de dados é usado para criar um esquema para a sua tabela. Você pode usar classificadores integrados ou definir o seu próprio. Você define os classificadores personalizados em uma operação separada, antes de definir os rastreamentos. O AWS Glue fornece classificadores integrados para inferir esquemas de arquivos comuns com formatos que incluem JSON, CSV e Apache Avro. Para ver a lista atual de classificadores integrados no AWS Glue, consulte Classificadores integrados no AWS Glue.

As tabelas de metadados que um crawler cria ficam contidas em um banco de dados quando você define um crawler. Se o seu crawler não especificar um banco de dados, suas tabelas serão colocadas no banco de dados padrão. Além disso, cada tabela possui uma coluna de classificação preenchida pelo primeiro classificador que reconheceu com sucesso o armazenamento de dados.

Se o arquivo rastreado estiver compactado, o crawler precisará fazer download dele para processá-lo. Quando um crawler é executado, ele interroga os arquivos para determinar seu formato e tipo de compactação e grava essas propriedades no Data Catalog. Alguns formatos de arquivo (por exemplo, Apache Parquet) permitem que você compacte partes do arquivo à medida que ele é gravado. Para esses arquivos, os dados compactados são um componente interno do arquivo e o AWS Glue não preenche a propriedade `compressionType` quando grava tabelas no Data Catalog. Por outro lado, se um arquivo inteiro for compactado por um algoritmo de compactação (por exemplo, gzip), a propriedade `compressionType` será preenchida quando as tabelas forem gravadas no Data Catalog.

O crawler gera os nomes das tabelas que ele cria. Os nomes das tabelas armazenadas no AWS Glue Data Catalog obedecem a estas regras:

- São permitidos somente caracteres alfanuméricos e sublinhados (_).
- Prefixos personalizados não podem conter mais do que 64 caracteres.
- O comprimento máximo do nome não pode ser superior a 128 caracteres. O crawler trunca nomes gerados para ajustá-los de acordo com o limite.
- Se forem encontrados nomes de tabelas duplicados, o crawler adicionará um sufixo de string hash a esse nome.

Se seu crawler for executado mais de uma vez (talvez em uma programação), ele procurará arquivos ou tabelas novos ou alterados no seu armazenamento de dados. A saída do crawler inclui novas tabelas e partições encontradas desde a execução anterior.

Quais armazenamentos de dados posso rastrear?

Os crawlers podem rastrear os seguintes armazenamentos de dados baseados em arquivos e baseados em tabelas.

Tipo de acesso que o crawler usa	Armazenamentos de dados
Cliente nativo	• Amazon Simple Storage Service (Amazon S3) • Amazon DynamoDB • Lago Delta 2.0.x • Apache Iceberg 1.5 • Apache Hudi 0.14
JDBC	Amazon Redshift Snowflake No Amazon Relational Database Service (Amazon RDS) ou externo ao Amazon RDS: • Amazon Aurora • MariaDB • Microsoft SQL Server • MySQL • Oracle • PostgreSQL
Cliente MongoDB	• MongoDB • MongoDB Atlas • Amazon DocumentDB (compatível com MongoDB)

> **Note**
>
> Atualmente, o AWS Glue não é compatível com crawlers para streams de dados.

Para armazenamentos de dados JDBC, MongoDB, MongoDB Atlas e Amazon DocumentDB (compatível com MongoDB), você deve especificar uma conexão do AWS Glue que o crawler possa usar para se conectar ao datastore. Para o Amazon S3, você pode especificar opcionalmente uma conexão do tipo Network (Rede). Uma conexão é um objeto do Data Catalog que armazena informações de conexão, como credenciais, URL, informações da Amazon Virtual Private Cloud e muito mais. Para ter mais informações, consulte Conectar a dados.

As seguintes versões de drivers são compatíveis com o crawler:

Produto	Driver compatível com crawler
PostgreSQL	42.2.1
Amazon Aurora	O mesmo que os drivers de crawler nativos
MariaDB	8.0.13
Microsoft SQL Server	6.1.0
MySQL	8.0.13
Oracle	11.2.2
Amazon Redshift	4.1
Snowflake	3.13.20
MongoDB	4.7.2
MongoDB Atlas	4.7.2

Veja a seguir observações sobre os vários armazenamentos de dados.

Amazon S3

Você pode escolher rastrear um caminho na sua conta ou em outra. Se todos os arquivos do Amazon S3 em uma pasta tiverem o mesmo esquema, o crawler criará uma tabela. Além disso, se o objeto do Amazon S3 for particionado, apenas uma tabela de metadados será criada e as informações da partição serão adicionadas ao Data Catalog para essa tabela.

Amazon S3 e Amazon DynamoDB

Os crawlers usam uma função do AWS Identity and Access Management (IAM) para conceder permissão de acesso aos armazenamentos de dados. A função que você transmite ao crawler precisa ter permissão para acessar os caminhos do Amazon S3 e as tabelas do Amazon DynamoDB que serão rastreados.

Amazon DynamoDB

Ao definir um crawler usando o console do AWS Glue, você especifica uma tabela do DynamoDB. Se você estiver usando a API do AWS Glue, pode especificar uma lista de tabelas. Você pode optar por rastrear apenas uma pequena amostra dos dados, para reduzir os tempos de execução do crawler.

Delta Lake

Para cada datastore Delta Lake, é necessário especificar como criar tabelas do Delta:

- Criar tabelas nativas: permite a integração com mecanismos de consulta compatíveis com consultas diretas ao log de transações do Delta. Para obter mais informações, consulte Consultar tabelas do Delta Lake.
- Criar tabelas de link simbólico: crie uma pasta `_symlink_manifest` com arquivos de manifesto particionados pelas chaves de partição com base nos parâmetros de configuração especificados.

Iceberg

Para cada datastore do Iceberg, você especifica um caminho do Amazon S3 que contém os metadados para as tabelas do Iceberg. Se o crawler descobrir metadados da tabela do Iceberg, ele os registrará no catálogo de dados. Você pode definir uma agenda para que o crawler mantenha as tabelas atualizadas.

Você pode definir esses parâmetros para o datastore:

- Exclusões: permite que você pule determinadas pastas.

- Profundidade máxima de travessia: define o limite de profundidade que o crawler pode percorrer no bucket do Amazon S3. A profundidade de travessia máxima padrão é 10 e a profundidade máxima que você pode definir é 20.

Hudi

Para cada datastore do Hudi, você especifica um caminho do Amazon S3 que contém os metadados para as tabelas do Hudi. Se o crawler descobrir metadados da tabela do Hudi, ele os registrará no catálogo de dados. Você pode definir uma agenda para que o crawler mantenha as tabelas atualizadas.

Você pode definir esses parâmetros para o datastore:

- Exclusões: permite que você pule determinadas pastas.
- Profundidade máxima de travessia: define o limite de profundidade que o crawler pode percorrer no bucket do Amazon S3. A profundidade de travessia máxima padrão é 10 e a profundidade máxima que você pode definir é 20.

> **Note**
>
> As colunas de timestamp tendo `millis` como tipos lógicos serão interpretadas como `bigint`, devido a uma incompatibilidade com o Hudi 0.13.1 e os tipos de timestamp. Uma resolução pode ser fornecida na próxima versão do Hudi.

As tabelas do Hudi são categorizadas da seguinte forma, com implicações específicas para cada uma delas:

- Copy on Write (CoW): os dados são armazenados em um formato colunar (Parquet) e cada atualização cria uma nova versão dos arquivos durante uma gravação.
- Merge on Read (MoR): os dados são armazenados usando uma combinação de formatos colunares (Parquet) e baseados em linha (Avro). As atualizações são registradas em arquivos delta baseados em linha e são compactadas conforme necessário para criar novas versões dos arquivos colunares.

Com conjuntos de dados CoW, sempre que há uma atualização para um registro, o arquivo que contém o registro é regravado com os valores atualizados. Com um conjunto de dados MoR, sempre que há uma atualização, o Hudi grava apenas a linha do registro alterado. MoR é mais adequado para cargas de trabalho com maior volume de gravações ou alterações e menor

volume de leituras. O tipo CoW é mais adequado para workloads com maior volume de leituras em dados que mudam com menos frequência.

O Hudi oferece três tipos de consulta para acessar os dados:

- Consultas de snapshot: consultas que veem o snapshot mais recente da tabela a partir de uma determinada ação de confirmação ou compactação. Para tabelas MoR, as consultas de snapshot expõem o estado mais recente da tabela mesclando os arquivos base e delta da fatia de arquivo mais recente no momento da consulta.
- Consultas incrementais: as consultas veem somente os novos dados gravados na tabela, a partir de uma determinada confirmação/compactação. Desse modo, os streams de alteração são fornecidos para habilitar pipelines de dados incrementais.
- Ler consultas otimizadas: para tabelas de MoR, as consultas veem os dados mais recentes compactados. Para tabelas CoW, as consultas veem os dados mais recentes confirmados.

Para tabelas Copy-On-Write, os crawlers criam uma única tabela no catálogo de dados com o serde ReadOptimized `org.apache.hudi.hadoop.HoodieParquetInputFormat`.

Para tabelas Merge-On-Read, o crawler cria duas tabelas no catálogo de dados para o mesmo local da tabela:

- Uma tabela com sufixo `_ro` que usa o serde ReadOptimized `org.apache.hudi.hadoop.HoodieParquetInputFormat`.
- Uma tabela com sufixo `_rt` que usa o serde RealTime permitindo consultas instantâneas: `org.apache.hudi.hadoop.realtime.HoodieParquetRealtimeInputFormat`.

MongoDB e Amazon DocumentDB (compatível com MongoDB)

O MongoDB versões 3.2 e posteriores são compatíveis. Você pode optar por rastrear apenas uma pequena amostra dos dados, para reduzir os tempos de execução do crawler.

Banco de dados relacional

A autenticação é realizada com um nome de usuário e senha do banco de dados. Dependendo do tipo de mecanismo de banco de dados, você pode escolher quais objetos são rastreados, como bancos de dados, esquemas e tabelas.

Snowflake

O crawler JDBC do Snowflake é compatível com crawling da tabela, da tabela externa, da visão e da visão materializada. A definição de visão materializada não será preenchida.

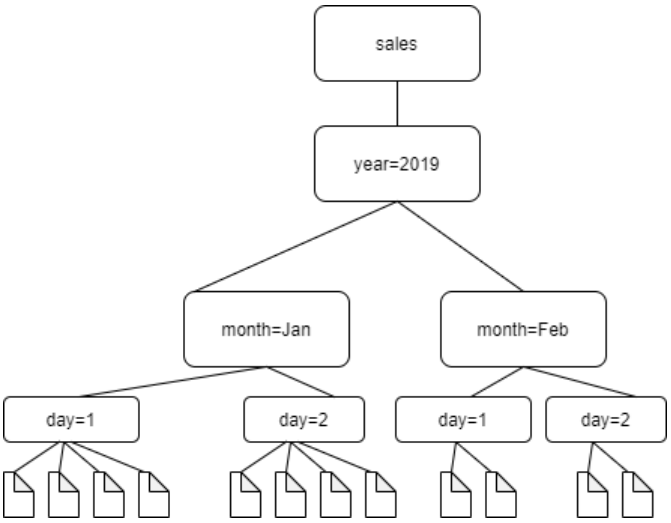
Em tabelas externas do Snowflake, o crawler só fará o rastreamento se apontar para um local do Amazon S3. Além do esquema da tabela, o crawler também rastreará o local, o formato do arquivo e a saída do Amazon S3 como parâmetros da tabela do Data Catalog. Observe que as informações de partição da tabela externa particionada não são preenchidas.

Atualmente, o ETL não é compatível com tabelas do Data Catalog criadas usando o crawler do Snowflake.

Como um crawler determina quando criar partições?

Quando um crawler do AWS Glue examina o Amazon S3 e detecta várias pastas em um bucket, ele determina a raiz de uma tabela na estrutura de pastas e quais pastas são partições de uma tabela. O nome da tabela é baseado no prefixo do Amazon S3 ou no nome da pasta. Você fornece um Include path (caminho de inclusão) que indica o nível da pasta a ser rastreada. Quando a maioria dos esquemas em um nível de pasta é semelhante, o crawler cria partições de uma tabela em vez de duas tabelas separadas. Para influenciar o crawler a criar tabelas separadas, adicione a pasta raiz de cada tabela como um repositório de dados separado ao definir o crawler.

Por exemplo, considere a seguinte estrutura de pastas do Amazon S3.



Os caminhos para as quatro pastas de nível mais baixo são os seguintes:

```
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
```

Suponha que o destino do crawler esteja definido em `Sales` e que todos os arquivos no `day=`*n* têm o mesmo formato (por exemplo, JSON, não criptografado) e têm os mesmos esquemas, ou esquemas muito semelhantes. O crawler criará uma única tabela com quatro partições, com chaves de partição `year`, `month` e `day`.

No exemplo a seguir, considere a seguinte estrutura do 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
```

Se os esquemas para arquivos em `table1` e `table2` forem semelhantes e um único armazenamento de dados estiver definido no crawler com Include path (Caminho de inclusão) `s3://bucket01/folder1/`, o crawler cria uma única tabela com duas colunas de chave de partição. A primeira coluna de chave de partição contém `table1` e `table2`, e a segunda coluna de chave de partição contém `partition1` a `partition3` para a partição da `table1` e `partition4` e `partition5` para a partição da `table2`. Para criar duas tabelas separadas, defina o crawler com dois armazenamentos de dados. Neste exemplo, defina o primeiro Include path (Caminho de inclusão) como `s3://bucket01/folder1/table1/`, e o segundo como `s3://bucket01/folder1/table2`.

> **Note**
>
> No Amazon Athena, cada tabela corresponde a um prefixo do Amazon S3 com todos os objetos nele. Se os objetos têm diferentes esquemas, o Athena não reconhece os objetos diferentes no mesmo prefixo como tabelas separadas. Isso pode acontecer se um crawler criar várias tabelas a partir do mesmo prefixo do Amazon S3. Isso pode levar a consultas no Athena que retornam zero resultados. Para que o Athena reconheça e consulte corretamente as tabelas, crie o crawler com um Include path (Caminho de inclusão) separado para cada esquema de tabela diferente na estrutura de pastas do Amazon S3. Para obter mais informações, consulte Práticas recomendadas ao usar o Athena com o AWS Glue e este artigo da Central de Conhecimento da AWS.

Pré-requisitos do crawler

O crawler assume as permissões da função do AWS Identity and Access Management (IAM) que você especificou ao defini-lo. Essa função do IAM precisa ter permissões para extrair dados do seu armazenamento de dados e gravar no Data Catalog. O console do AWS Glue lista somente as funções do IAM com uma política de confiança anexada para o serviço da entidade principal AWS Glue. No console, você também pode criar uma função do IAM com uma política do IAM para acessar o armazenamento de dados do Amazon S3 que é acessado pelo crawler. Para obter mais informações sobre como fornecer funções ao AWS Glue, consulte Políticas baseadas em identidade para Glue AWS.

> **Note**
>
> Para fazer o crawling de um armazenamento de dados do Delta Lake, é necessário ter permissões de leitura/gravação para o local do Amazon S3.

Para o crawler, você pode criar uma função e anexar as seguintes políticas:

- A política `AWSGlueServiceRole` gerenciada pela AWS, que concede as permissões necessárias no Data Catalog
- Uma política em linha que concede permissões na origem dos dados.
- Uma política em linha que concede permissões `iam:PassRole` no perfil.

Uma abordagem mais rápida é permitir que o assistente do crawler do console do AWS Glue crie uma função para você. A função que ele cria é especificamente para o crawler e inclui a política `AWSGlueServiceRole` gerenciada pela AWS e mais a política em linha necessária para a origem dos dados especificada.

Se você especificar uma função existente para um crawler, certifique-se de que ela inclua a política `AWSGlueServiceRole` ou equivalente (ou uma versão dessa política com um escopo reduzido), além das políticas em linha necessárias. Por exemplo, para um armazenamento de dados do Amazon S3, a política em linha seria, no mínimo, a seguinte:

```
{
   "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket/object*"
            ]
        }
    ]
}
```

Para um armazenamento de dados do Amazon DynamoDB, a política seria, no mínimo, a seguinte:

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

E ainda, se o crawler ler os dados do Amazon S3 criptografados pelo AWS Key Management Service (AWS KMS), a função do IAM AWS KMS deverá ter permissão de descriptografia na chave do . Para ter mais informações, consulte Etapa 2: criar um perfil do IAM para o AWS Glue.

Configurar um crawler

Um crawler acessa seu armazenamento de dados, extrai metadados e cria definições de tabela do AWS Glue Data Catalog. O painel Crawlers no console do AWS Glue lista todos os crawlers que você cria. A lista exibe status e métricas da última execução do seu crawler.

> **Note**
>
> Se você optar por trazer suas próprias versões do driver JDBC, os crawlers do AWS Glue consumirão recursos em trabalhos do AWS Glue e buckets do Amazon S3 para garantir que o driver fornecido seja executado em seu ambiente. O uso adicional de recursos será refletido em sua conta. Além disso, fornecer seu próprio driver JDBC não significa que o crawler seja capaz de aproveitar todos os atributos do driver. Os drivers estão limitados às propriedades descritas em Adicionar uma conexão do AWS Glue.

Para configurar um crawler

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/. Escolha Crawlers no painel de navegação.
2. Escolha Adicionar crawler e siga as instruções no assistente Adicionar crawler. O assistente orientará você pelas etapas necessárias para criar um crawler. Se quiser adicionar classificadores personalizados para definir o esquema, consulte Adicionar classificadores a um crawler no AWS Glue.

Etapa 1: configurar propriedades do crawler

Insira um nome e uma descrição (opcional) para o crawler. Opcionalmente, você pode marcar o crawler com uma Tag key (Chave de tag) e um Tag value (Valor de tag) opcional. Depois de serem criadas, as chaves de tag são somente leitura. Use as tags em alguns recursos para ajudar a organizá-los e identificá-los. Para obter mais informações, consulte tags da AWS no AWS Glue.

Nome

O nome pode conter letras (A a Z), números (0 a 9), hifens (-) ou sublinhados (_), e pode ter até 255 caracteres.

Descrição

A descrição pode ter até 2048 caracteres.

Tags

Use tags para organizar e identificar os recursos. Para mais informações, consulte:

- Etiquetas da AWS no AWS Glue

Etapa 2: escolher as fontes de dados e os classificadores

Configuração da fonte de dados

Selecione a opção apropriada para Os dados já estão mapeados em tabelas do AWS Glue?. Escolha "Ainda não" ou "Sim". A opção "Ainda não" está selecionada por padrão.

O crawler pode acessar armazenamentos de dados diretamente como a fonte do crawl ou pode usar tabelas existentes no Data Catalog como a fonte. Se o crawler usa tabelas de catálogos existentes, ele rastreia os armazenamentos de dados que são especificados por essas tabelas do catálogo.

- Se ainda não estiverem: selecione uma ou mais fontes de dados a serem rastreadas. Um crawler pode rastrear vários armazenamentos de dados de tipos diferentes (Amazon S3, JDBC e outros).

  Você só pode configurar um datastore por vez. Depois de fornecer as informações de conexão, incluir caminhos e excluir padrões, você terá a opção de adicionar outro datastore.
- Sim: selecione tabelas existentes do AWS Glue Data Catalog. As tabelas do catálogo especificam os armazenamentos de dados a serem rastreados. O crawler pode rastrear somente tabelas do catálogo em uma única execução; ele não pode misturar outros tipos de origem.

  Um motivo comum para especificar uma tabela de catálogo como fonte é quando você cria a tabela manualmente (pois você já conhece a estrutura do datastore) e deseja que um crawler a mantenha atualizada, incluindo a adição de novas partições. Para ver uma discussão de outros motivos, consulte Atualizar tabelas do Data Catalog criadas manualmente usando crawlers.

  Quando você especifica tabelas existentes como o tipo de fonte do crawler, as seguintes condições se aplicam:

  - O nome do banco de dados é opcional.
  - Somente tabelas de catálogo que especificam armazenamentos de dados do Amazon S3 ou Amazon DynamoDB são permitidas.
  - Nenhuma tabela de catálogo nova é criada quando o crawler é executado. As tabelas existentes são atualizadas conforme necessário, incluindo a adição de novas partições.
  - Os objetos excluídos encontrados nos armazenamentos de dados são ignorados; nenhuma tabela de catálogo é excluída. Em vez disso, o crawler grava uma mensagem de log. (`SchemaChangePolicy.DeleteBehavior=LOG`)

- A opção de configuração do crawler para criar um único esquema para cada caminho do Amazon S3 é habilitada por padrão e não pode ser desabilitada. (`TableGroupingPolicy=CombineCompatibleSchemas`) Para obter mais informações, consulte Como criar um esquema único para cada caminho de inclusão do Amazon S3.
- Você não pode misturar as tabelas de catálogo como uma fonte com nenhum outro tipo de fonte (por exemplo, Amazon S3 ou Amazon DynamoDB).

Fontes de dados

Selecione ou adicione a lista de fontes de dados a serem varridas pelo crawler.

(Opcional) Se você escolher JDBC como fonte de dados, poderá usar seus próprios drivers JDBC ao especificar o acesso da conexão em que as informações do driver são armazenadas.

Incluir o caminho

Ao avaliar o que incluir ou excluir em um crawling, o crawler começa pelo caminho de inclusão obrigatório. Para o Amazon S3, o MongoDB, o MongoDB Atlas, o Amazon DocumentDB (compatível com MongoDB) e armazenamentos de dados relacionais, você deve especificar um caminho de inclusão.

Para um datastore do Amazon S3

Escolha se deseja especificar um caminho na sua conta ou em uma conta diferente e depois procure e escolha um caminho do Amazon S3.

Para armazenamentos de dados do Amazon S3, a sintaxe de inclusão de caminho é `bucket-name/folder-name/file-name.ext`. Para rastrear todos os objetos em um bucket, você especifica apenas o nome do bucket no caminho de inclusão. O padrão de exclusão é relativo ao caminho de inclusão

Para um datastore do Delta Lake

Especifique um ou mais caminhos do Amazon S3 para tabelas do Delta, como s3://*bucket*/*prefixo*/*objeto*.

Para um datastore do Iceberg ou do Hudi

Especifique um ou mais caminhos do Amazon S3 que contenham pastas com metadados da tabela do Iceberg ou do Hudi, como s3://*bucket*/*prefixo*.

Para um datastore, a pasta do Hudi pode estar localizada em uma pasta secundária da pasta raiz. O crawler examinará todas as pastas abaixo de um caminho para uma pasta do Hudi.

Para um datastore no JDBC

Insira *<banco de dados>*/*<esquema>*/*<tabela>* ou *<banco de dados>*/*<tabela>*, dependendo do produto do banco de dados. Oracle Database e MySQL não oferecem suporte a esquema no caminho. Você pode substituir o caractere percentual (%) por *<esquema>* ou *<tabela>*. Por exemplo, para um banco de dados Oracle com um identificador de sistema (SID) `orcl`, informe `orcl/%` para importar todas as tabelas às quais o usuário nomeado na conexão tem acesso.

> **Important**
>
> Este campo diferencia letras maiúsculas de minúsculas.

Para um datastore MongoDB, MangoDB Atlas ou Amazon DocumentDB

Digite *database*/*collection* (banco de dados/coleção).

Para o MongoDB, o MongoDB Atlas e o Amazon DocumentDB (compatível com MongoDB), a sintaxe é `database/collection`.

Para armazenamentos de dados JDBC, a sintaxe é `database-name/schema-name/table-name` ou `database-name/table-name`. A sintaxe depende de o mecanismo de banco de dados ser compatível com esquemas dentro de um banco de dados. Por exemplo, para mecanismos de banco de dados, como MySQL ou Oracle, não especifique um `schema-name` em seu caminho de inclusão. Você pode substituir o sinal de porcentagem (`%`) por um esquema ou tabela no caminho de inclusão para representar todos os esquemas ou todas as tabelas em um banco de dados. Não é possível substituir o sinal de porcentagem (`%`) do banco de dados no caminho de inclusão.

Profundidade transversal máxima (somente para armazenamentos de dados do Iceberg ou do Hudi)

A profundidade máxima de caminhos do Amazon S3 que o crawler pode percorrer para descobrir a pasta de metadados do Iceberg ou do Hudi no caminho do Amazon S3. O objetivo desse parâmetro é limitar o run time do crawler. O valor padrão é 10 e o máximo é 20.

Padrões de exclusão

Eles permitem que você exclua determinados arquivos ou tabelas do rastreamento. O caminho de exclusão é relativo ao caminho de inclusão. Por exemplo, para excluir uma tabela no seu datastore JDBC, digite o nome dela no caminho de exclusão.

Um crawler se conecta a um datastore JDBC usando uma conexão do AWS Glue que contém uma string de conexão de URI do JDBC. O crawler só tem acesso a objetos no mecanismo de banco de dados usando o nome de usuário e a senha do JDBC na conexão do AWS Glue. O crawler só pode criar tabelas que ele pode acessar por meio da conexão JDBC. Depois que o crawler acessa o mecanismo de banco de dados com o URI do JDBC, o caminho de inclusão é usado para determinar quais tabelas no mecanismo de banco de dados são criadas no Data Catalog. Por exemplo, com o MySQL, se você especificar um caminho de inclusão de `MyDatabase/%`, todas as tabelas em `MyDatabase` serão criadas no Data Catalog. Ao acessar o Amazon Redshift, se você especificar um caminho de inclusão de `MyDatabase/%`, todas as tabelas dentro de todos os esquemas para o banco de dados `MyDatabase` serão criadas no Data Catalog. Se você especificar um caminho de inclusão de `MyDatabase/MySchema/%`, todas as tabelas no banco de dados `MyDatabase` e o esquema `MySchema` serão criados.

Depois de especificar um caminho de inclusão, você pode excluir objetos do rastreamento que o seu caminho de inclusão incluiria especificando um ou mais padrões de exclusão `glob` no estilo Unix. Esses padrões são aplicados ao seu caminho de inclusão para determinar quais objetos serão excluídos. Esses padrões também são armazenados como uma propriedade de tabelas criadas pelo crawler. AWS Glue As extensões PySpark, como `create_dynamic_frame.from_catalog`, leem as propriedades da tabela e excluem os objetos definidos pelo padrão de exclusão.

O AWS Glue oferece suporte aos seguintes tipos de padrão `glob` no padrão de exclusão.

Padrão de exclusão	Descrição
`*.csv`	Corresponde a um caminho do Amazon S3 que representa um nome de objeto na pasta atual que termina em `.csv`
`*.*`	Corresponde a todos os nomes de objetos que contêm um ponto
`*.{csv,avro}`	Corresponde aos nomes de objetos que terminam em `.csv` ou `.avro`
`foo.?`	Corresponde aos nomes de objetos que começam com `foo.` seguidos por uma extensão de caractere único

Padrão de exclusão	Descrição
`myfolder/*`	Corresponde a objetos em um nível de subpasta de `myfolder`, como `/myfolder/mysource`
`myfolder/*/*`	Corresponde a objetos em dois níveis de subpastas de `myfolder`, como `/myfolder/mysource/data`
`myfolder/**`	Corresponde a objetos em todas as subpastas `myfolder`, por exemplo, `/myfolder/mysource/mydata` e `/myfolder/mysource/data`
`myfolder**`	Corresponde a subpasta `myfolder` bem como aos arquivos abaixo de `myfolder`, por exemplo, `/myfolder` e `/myfolder/mydata.txt`
`Market*`	Corresponde a tabelas em um banco de dados JDBC com nomes que começam com `Market`, como `Market_us` e `Market_fr`

O AWS Glue interpreta os padrões de exclusão `glob` da seguinte forma:

- O caractere de barra (`/`) é o delimitador que separa as chaves do Amazon S3 em uma hierarquia de pastas.
- O caractere de asterisco (`*`) corresponde a zero ou mais caracteres de um componente de nome sem cruzar limites da pasta.
- Um asterisco duplo (`**`) corresponde a zero ou mais caracteres cruzando limites de pasta ou esquema.
- O caractere de ponto de interrogação (`?`) corresponde a exatamente um caractere de um componente de nome.
- O caractere de barra invertida (`\`) é usado para realizar o escape de caracteres que não podem ser interpretados como caracteres especiais. A expressão `\\` corresponde a uma única barra invertida, e `\{` corresponde a uma chave esquerda.

- Os colchetes `[ ]` criam uma expressão que corresponde a um único caractere de um componente de nome fora de um conjunto de caracteres. Por exemplo, `[abc]` corresponde a `a`, `b` ou `c`. O hífen (`-`) pode ser usado para especificar um intervalo, portanto, `[a-z]` especifica um intervalo correspondente de `a` a `z` (de inclusão). Esses formatos podem ser combinados, portanto, `[abce-g]` corresponde a `a`, `b`, `c`, `e`, `f` ou `g`. Se o caractere depois do colchete (`[`) for um ponto de exclamação (`!`), a expressão de colchetes será negada. Por exemplo, `[!a-c]` corresponde a qualquer caractere, exceto `a`, `b` ou `c`.

  Em uma expressão de colchetes, os caracteres `*`, `?` e `\` se correspondentes entre si. O caractere de hífen (`-`) corresponde a si mesmo se for o primeiro caractere dentro de colchetes ou o primeiro caractere após `!` quando você estiver fazendo uma negação.
- As chaves (`{ }`) englobam um grupo de subpadrões, e o grupo será correspondente se houver correspondência com qualquer subpadrão no grupo. Um caractere de vírgula (`,`) é usado para separar os subpadrões. Grupos não podem ser aninhados.
- Os caracteres de ponto inicial ou final em nomes de arquivos são tratados como caracteres normais nas operações de correspondência. Por exemplo, o padrão de exclusão `*` corresponde ao nome do arquivo `.hidden`.

Example Padrões de exclusão do Amazon S3

Cada padrão de exclusão é avaliado em comparação com o caminho de inclusão. Por exemplo, digamos que você tenha a seguinte estrutura de diretórios do Amazon S3:

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

Dado o caminho de inclusão `s3://mybucket/myfolder/`, estes são alguns resultados de amostra para padrões de exclusão:

Padrão de exclusão	Resultados
`departments/**`	Exclui todos os arquivos e pastas abaixo de `departments` e inclui a pasta `employees` e seus respectivos arquivos
`departments/market*`	Exclui `market-us.json` , `market-emea.json` e `market-ap.json`
`**.csv`	Exclui todos os objetos abaixo de `myfolder` com nomes que terminam em `.csv`
`employees/*.csv`	Exclui todos os arquivos `.csv` na pasta `employees`

Example Excluir um subconjunto de partições do Amazon S3

Suponha que seus dados sejam particionados por dia, de modo que cada dia em um ano esteja em uma partição diferente do Amazon S3. Para janeiro de 2015, há 31 partições. Agora, para rastrear dados apenas na primeira semana de janeiro, é necessário excluir todas as partições, exceto os dias 1 a 7:

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

Analise as partes deste padrão glob. A primeira parte, `2015/01/{[!0],0[8-9]}**`, exclui todos os dias que não começam com um "0", além dos dias 08 e 09 do mês 01 no ano 2015. Observe que "**" é usado como o sufixo para o padrão de número do dia e ultrapassa os limites de pasta para as pastas de nível inferior. Se "*" for usado, os níveis mais baixos da pasta não serão excluídos.

A segunda parte, `2015/0[2-9]/**`, exclui os dias do mês 02 ao mês 09, no ano 2015.

A terceira parte, `2015/1[0-2]/**`, exclui os dias do mês 10, 11 e 12, no ano 2015.

Example Padrões de exclusão do JDBC

Suponha que você esteja realizando crawling de um banco de dados JDBC com a seguinte estrutura de esquema:

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

Dado o caminho de inclusão `MyDatabase/MySchema/%`, estes são alguns resultados de amostra para padrões de exclusão:

Padrão de exclusão	Resultados
`HR*`	Exclui as tabelas com nomes que começam com `HR`
`Market_*`	Exclui as tabelas com nomes que começam com `Market_`
`**_Table`	Exclui todas as tabelas com nomes que terminam em `_Table`

Parâmetros adicionais de fonte do crawler

Cada tipo de fonte requer um conjunto diferente de parâmetros adicionais. A seguir, uma lista incompleta:

Conexão

Selecione ou adicione uma conexão do AWS Glue. Para obter informações sobre conexões, consulte Conectar a dados.

Metadados adicionais - opcionais (para armazenamentos de dados JDBC)

Selecione propriedades de metadados adicionais para o crawler rastrear.

- Comentários: rastreie os comentários associados no nível da tabela e no nível da coluna.
- Tipos brutos: mantenha os tipos de dados brutos das colunas da tabela em metadados adicionais. Como comportamento padrão, o crawler traduz os tipos de dados brutos em tipos compatíveis com o Hive.

Nome da classe do driver JDBC: opcional (para armazenamentos de dados JDBC)

Digite um nome de classe de driver JDBC personalizado para que o crawler se conecte à fonte de dados:

- 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

Caminho do S3 para o driver JDBC: opcional (para armazenamentos de dados JDBC)

Escolha um caminho existente do Amazon S3 para um arquivo `.jar`. É nesse local que o arquivo `.jar` será armazenado quando um driver JDBC personalizado for usado para o crawler se conectar à fonte de dados.

Habilitar a amostragem de dados (somente para o armazenamentos de dados Amazon DynamoDB, MongoDB e MongoDB Atlas e Amazon DocumentDB)

Selecione se deseja rastrear somente uma amostra de dados. Se essa opção não for selecionada, a tabela inteira será rastreada. A verificação de todos os registros pode levar muito tempo quando a tabela não é de throughput alto.

Criar tabelas para consultas (somente para armazenamentos de dados do Delta Lake)

Selecione como você deseja criar as tabelas do Delta Lake:

- Create Native tables (Criar tabelas nativas): permite a integração com mecanismos de consulta compatíveis com consulta direta ao log de transações do Delta.
- Create Symlink tables (Criar tabelas de link simbólico): Crie uma pasta manifesto de link simbólico com arquivos de manifesto particionados pelas chaves de partição, com base nos parâmetros de configuração especificados.

Scanning rate (Taxa de varredura): opcional (somente para armazenamentos de dados do DynamoDB)

Especifique a porcentagem das unidades de capacidade de leitura da tabela do DynamoDB a serem usadas pelo crawler. Unidades de capacidade de leitura é um termo definido pelo DynamoDB e é um valor numérico que atua como limitador de taxa para o número de leituras que podem ser executadas nessa tabela por segundo. Insira um valor entre 0,1 e 1,5. Se não for especificado, o padrão será de 0,5% para tabelas provisionadas e 1/4 da capacidade máxima configurada para tabelas sob demanda. Observe que somente o modo de capacidade provisionada deve ser usado com crawlers do AWS Glue.

> **Note**
>
> Para armazenamentos de dados do DynamoDB, defina o modo de capacidade provisionada para processar leituras e gravações em suas tabelas. O crawler do AWS Glue não deve ser usado com o modo de capacidade sob demanda.

Network connection (Conexão de rede): opcional (somente para armazenamentos de dados do Amazon S3)

Opcionalmente, inclua uma conexão de rede para usar com esse destino do Amazon S3. Observe que cada crawler é limitado a uma conexão de rede, portanto, qualquer outro destino do Amazon S3 também usará a mesma conexão (ou nenhuma, se deixada em branco).

Para obter informações sobre conexões, consulte Conectar a dados.

Amostra apenas um subconjunto de arquivos e tamanho da amostra (somente para armazenamentos de dados do Amazon S3)

Especifique o número de arquivos em cada pasta de folha a serem rastreados durante o crawling de arquivos de amostra em um conjunto de dados. Quando esse recurso é ativado, em vez de realizar o crawling em todos os arquivos neste conjunto de dados, o crawler seleciona aleatoriamente alguns arquivos em cada pasta de folha para rastrear.

O crawler de amostragem é mais adequado para clientes que têm conhecimento prévio sobre seus formatos de dados e sabem que os esquemas em suas pastas não são alterados. Ativar esse recurso reduzirá significativamente o runtime do crawler.

Um valor válido é um número inteiro entre 1 e 249. Se não for especificado, todos os arquivos serão rastreados.

Execuções subsequentes do crawler

Esse campo é um campo global que afeta todas as fontes de dados do Amazon S3.

- Crawl all sub-folders (Rastrear todas as subpastas): rastrear todas as pastas novamente a cada rastreamento subsequente.
- Crawl new sub-folders only (Rastrear somente novas subpastas): somente as pastas do Amazon S3 que foram adicionadas desde o último rastreamento serão rastreadas. Se os esquemas forem compatíveis, novas partições serão adicionadas às tabelas existentes. Para ter mais informações, consulte the section called “Crawls incrementais para adicionar novas partições”.
- Crawl based on events (Rastreamento com base em eventos): depende dos eventos do Amazon S3 para controlar quais pastas rastrear. Para ter mais informações, consulte the section called “Acelerar crawls usando notificações de eventos do Amazon S3”.

Custom classifiers (Classificadores personalizados) - opcionais

Defina classificadores personalizados antes de definir crawlers. Um classificador verifica se determinado arquivo está em um formato que pode ser processado pelo crawler. Se estiver, o classificador cria um esquema na forma de um objeto `StructType` que corresponde a esse formato de dados.

Para ter mais informações, consulte Adicionar classificadores a um crawler no AWS Glue.

Etapa 3: Defina as configurações de segurança

IAM role (Perfil do IAM)

O crawler assume essa função. Ele deve ter permissões para a política `AWSGlueServiceRole` gerenciada pela AWS. Para fontes do Amazon S3 e do DynamoDB, ele também deve ter permissões para acessar o datastore. Se o crawler ler os dados do Amazon S3 criptografados pelo AWS Key Management Service (AWS KMS), a função do IAM deverá ter permissão de descriptografia na chave do AWS KMS.

Para um datastore do Amazon S3, permissões adicionais anexadas à função seriam semelhantes às seguintes:

```
{
   "Version": "2012-10-17",
```

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

Para um datastore do Amazon DynamoDB, permissões adicionais anexadas à função seriam semelhantes às seguintes:

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

Para adicionar seu próprio driver JDBC, permissões adicionais precisam ser adicionadas.

- Conceda permissões para as seguintes ações: `CreateJob`, `DeleteJob`, `GetJob`, `GetJobRun`, `StartJobRun`.
- Conceda permissões para as ações do Amazon S3: `s3:DeleteObjects`, `s3:GetObject`, `s3:ListBucket`, `s3:PutObject`.

> **Note**
>
> `s3:ListBucket` não é necessário se a política de buckets do Amazon S3 estiver desabilitada.

- Conceda acesso de entidade principal de serviço a bucket/pasta na política do Amazon S3.

Exemplo de política do 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"
            ]
        }
    ]
}
```

O AWS Glue cria as pastas a seguir (`_crawler` e `_glue_job_crawler`) no mesmo nível do driver JDBC no bucket do Amazon S3. Por exemplo, se o caminho do driver for `<s3-path/driver_folder/driver.jar>`, as seguintes pastas serão criadas se ainda não existirem:

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

Opcionalmente, você pode adicionar uma configuração de segurança para um crawler para especificar opções de criptografia em repouso.

Para obter mais informações, consulte Etapa 2: criar um perfil do IAM para o AWS Glue e Gerenciamento de identidade e acesso para AWS Glue.

Lake Formation configuration (Configuração do Lake Formation) - opcional

Permita que o crawler use credenciais do Lake Formation para fazer crawling na fonte de dados.

Marcar Use Lake Formation credentials for crawling S3 data source (Usar credenciais do Lake Formation para rastrear a fonte de dados do S3) permitirá que o crawler use as credenciais do Lake Formation para rastrear a fonte de dados. Se a fonte de dados pertencer a outra conta, você deve fornecer a ID da conta registrada. Senão, o crawler rastreará somente as fontes de dados associadas à conta. Aplicável somente às fontes de dados do Amazon S3 e do Data Catalog.

Configuração de segurança - opcional

As configurações incluem configurações de segurança. Para mais informações, consulte:

- Criptografar dados gravados pelo AWS Glue

> **Note**
>
> Depois que uma configuração de segurança for definida em um crawler, você poderá alterá-la, mas não poderá removê-la. Para reduzir o nível de segurança em um crawler, defina explicitamente o recurso de segurança como `DISABLED` dentro da sua configuração ou crie um novo crawler.

Etapa 4: configurar a saída e o agendamento

Configuração da saída

As opções incluem como o crawler deve lidar com alterações detectadas do esquema, com a exclusão de objetos no datastore e muito mais. Para ter mais informações, consulte Personalizar o comportamento do Crawler.

Programação do crawler

Você pode executar o crawler sob demanda ou definir uma agenda baseada em hora para os crawlers e os trabalhos no AWS Glue. A definição dessas programações usa a sintaxe cron semelhante à do Unix. Para ter mais informações, consulte Programar um crawler do AWS Glue.

Etapa 5: analisar e criar

Revise as configurações do crawler que você definiu e crie o crawler.

Adicionar classificadores a um crawler no AWS Glue

Um classificador lê os dados em uma armazenamento de dados. Se reconhecer o formato dos dados, ele gerará um esquema. O classificador também retornará um número de certeza para indicar o nível de certeza referente ao reconhecimento do formato.

O AWS Glue fornece um conjunto de classificadores integrados, mas você também pode criar classificadores personalizados. O AWS Glue invoca classificadores personalizados primeiro, na ordem especificada na definição do crawler. Dependendo dos resultados retornados dos classificadores personalizados, o AWS Glue também poderá invocar classificadores integrados. Se um classificador retornar `certainty=1.0` durante o processamento, ele indicará que está 100% certo sobre a criação do esquema correto. Em seguida, o AWS Glue usa a saída desse classificador.

Se nenhum classificador retornar `certainty=1.0`, o AWS Glue usará a saída do classificador que tiver a maior certeza. Se nenhum classificador retornar uma certeza maior que `0.0`, o AWS Glue retornará a string de classificação padrão `UNKNOWN`.

Quando devo usar um classificador?

Você deve usar classificadores ao fazer crawling em um armazenamento de dados para definir tabelas de metadados no AWS Glue Data Catalog. Você pode configurar seu crawler com um conjunto de classificadores ordenados. Quando o crawler invoca um classificador, o classificador determina se os dados são reconhecidos. Se o classificador não puder reconhecer os dados ou não estiver 100% certo, o crawler chamará o próximo classificador na lista para tentar reconhecer os dados.

Para obter mais informações sobre como criar um classificador usando o console do AWS Glue, consulte Trabalhar com classificadores no console do AWS Glue.

Classificadores personalizados

A saída de um classificador inclui uma string que indica a classificação ou o formato do arquivo (por exemplo, `json`) e o esquema dele. Para classificadores personalizados, você define a lógica de criação do esquema com base no tipo de classificador. Os tipos de classificadores incluem a definição de esquemas com base em padrões grok, tags XML e caminhos JSON.

Se você alterar uma definição de classificador, os dados que foram rastreados anteriormente usando o classificador não serão reclassificados. Um crawler monitora os dados rastreados anteriormente. Novos dados são classificados com o classificador atualizado, o que pode resultar em um esquema atualizado. Se o esquema de seus dados evoluiu, atualize o classificador para considerar quaisquer alterações de esquema quando o crawler for executado. Para reclassificar dados para corrigir um classificador incorreto, crie um novo crawler com o classificador atualizado.

Para obter mais informações sobre como criar classificadores personalizados no AWS Glue, consulte Gravar classificadores personalizados.

> **Note**
>
> Se seu formato de dados for reconhecido por um dos classificadores integrados, você não precisará criar um classificador personalizado.

Classificadores integrados no AWS Glue

O AWS Glue fornece classificadores integrados para vários formatos, incluindo JSON, CSV, logs da web e diversos sistemas de banco de dados.

Se o AWS Glue não encontrar um classificador personalizado adequado para o formato de dados de entrada com 100% de certeza, ele invocará os classificadores integrados na ordem exibida na tabela a seguir. Os classificadores integrados retornam um resultado para indicar se o formato é correspondente (`certainty=1.0`) ou não (`certainty=0.0`). O primeiro classificador que tiver `certainty=1.0` fornecerá a string de classificação e o esquema para uma tabela de metadados no Data Catalog.

Tipo de classificador	String de classificação	Observações
Apache Avro	`avro`	Lê o esquema no início do arquivo para determinar o formato.
Apache ORC	`orc`	Lê os metadados do arquivo para determinar o formato.
Apache Parquet	`parquet`	Lê o esquema no final do arquivo para determinar o formato.

Tipo de classificador	String de classificação	Observações
JSON	`json`	Lê o início do arquivo para determinar o formato.
JSON binário	`bson`	Lê o início do arquivo para determinar o formato.
XML	`xml`	Lê o início do arquivo para determinar o formato. O AWS Glue determina o esquema de tabela com base em tags XML no documento. Para obter informações sobre como criar um classificador XML personalizado para especificar linhas no documento, consulte Gravar classificadores XML personalizados.
Amazon Ion	`ion`	Lê o início do arquivo para determinar o formato.
Log do Apache combinado	`combined_apache`	Determina os formatos de log por meio de um padrão grok.
Log do Apache	`apache`	Determina os formatos de log por meio de um padrão grok.
Log de kernel do Linux	`linux_kernel`	Determina os formatos de log por meio de um padrão grok.
Log da Microsoft	`microsoft_log`	Determina os formatos de log por meio de um padrão grok.
Log do Ruby	`ruby_logger`	Lê o início do arquivo para determinar o formato.
Log do Squid 3.x	`squid`	Lê o início do arquivo para determinar o formato.
Log de monitoramento do Redis	`redismonlog`	Lê o início do arquivo para determinar o formato.
Log do Redis	`redislog`	Lê o início do arquivo para determinar o formato.

Tipo de classificador	String de classificação	Observações
CSV	`csv`	Verifica os seguintes delimitadores: vírgula (,), pipe (|), tabulação (\t), ponto e vírgula (;) e Ctrl+A (\u0001). Ctrl+A é o caractere de controle Unicode para `Start Of Heading`.
Amazon Redshift	`redshift`	Usa conexão JDBC para importar metadados.
MySQL	`mysql`	Usa conexão JDBC para importar metadados.
PostgreSQL	`postgresql`	Usa conexão JDBC para importar metadados.
Banco de dados Oracle	`oracle`	Usa conexão JDBC para importar metadados.
Microsoft SQL Server	`sqlserver`	Usa conexão JDBC para importar metadados.
Amazon DynamoDB	`dynamodb`	Lê dados da tabela do DynamoDB.

Os arquivos nos seguintes formatos compactados podem ser classificados:

- (com suporte para arquivos ZIP contendo apenas um arquivo). Observe que o formato Zip não tem bom suporte em outros serviços (por causa do arquivamento).
- BZIP
- GZIP
- LZ4
- Snappy (compatível com os formatos Snappy Standard e Snappy de Hadoop nativo)

Classificador CSV integrado

O classificador CSV integrado analisa o conteúdo do arquivo CSV para determinar o esquema de uma tabela do AWS Glue. Esse classificador verifica os seguintes delimitadores:

- Vírgula (,)

- Barra vertical (|)
- Tabulação (\t)
- Ponto e vírgula (;)
- Ctrl+A (\u0001)

  Ctrl+A é o caractere de controle Unicode para `Start Of Heading`.

Para ser classificado como CSV, o esquema da tabela deve ter pelo menos duas colunas e duas linhas de dados. O classificador CSV usa um número de heurística para determinar se um cabeçalho está presente em um determinado arquivo. Se o classificador não puder determinar um cabeçalho a partir da primeira linha de dados, os cabeçalhos das colunas serão exibidos como `col1`, `col2`, `col3` e assim por diante. O classificador CSV integrado determina se é necessário inferir um cabeçalho avaliando as seguintes características do arquivo:

- Cada coluna em um possível cabeçalho é analisada como um tipo de dados STRING.
- Exceto pela última coluna, todas as colunas em um possível cabeçalho têm conteúdo com menos de 150 caracteres. Para permitir um delimitador final, a última coluna pode ficar vazia em todo o arquivo.
- Cada coluna em um possível cabeçalho deve atender aos requisitos do AWS Glue `regex` para um nome de coluna.
- A linha de cabeçalho deve ser suficientemente diferente das linhas de dados. Para determinar isso, uma ou mais linhas devem ser analisadas como diferentes do tipo STRING. Se todas as colunas forem do tipo STRING, a primeira linha de dados não será suficientemente diferente das linhas subsequentes a serem usadas como cabeçalho.

> **Note**
>
> Se o classificador CSV integrado não criar sua tabela do AWS Glue como você deseja, é possível usar uma das seguintes alternativas:
>
> - Altere os nomes das colunas no Data Catalog, defina `SchemaChangePolicy` como LOG e a configuração de saída da partição como `InheritFromTable` para futuras execuções do crawler.
> - Crie um classificador grok personalizado para analisar os dados e atribuir as colunas desejadas.

- O classificador CSV integrado cria tabelas fazendo referência ao `LazySimpleSerDe` como a biblioteca de serialização, que é uma boa opção para inferência de tipos. No entanto, se os dados CSV contiverem strings entre aspas, edite a definição da tabela e altere a biblioteca SerDe para `OpenCSVSerDe`. Ajuste os tipos inferidos para STRING, defina `SchemaChangePolicy` como LOG e defina a configuração de saída das partições como `InheritFromTable` para futuras execuções do crawler. Para mais informações sobre bibliotecas SerDe, consulte Referência de SerDe no Manual do usuário do Amazon Athena.

Gravar classificadores personalizados

Você pode fornecer um classificador personalizado para classificar seus dados no AWS Glue. Você pode criar um classificador personalizado usando um padrão grok, uma tag XML, JavaScript Object Notation (JSON) ou valores separados por vírgula (CSV). Um crawler do AWS Glue chama um classificador personalizado. Se o classificador reconhecer os dados, ele retornará a classificação e o esquema dos dados ao crawler. Você precisará definir um classificador personalizado se seus dados não coincidirem com os classificadores integrados ou se você quiser personalizar as tabelas criadas pelo crawler.

Para obter mais informações sobre como criar um classificador usando o console do AWS Glue, consulte Trabalhar com classificadores no console do AWS Glue.

O AWS Glue executa os classificadores personalizados antes dos classificadores integrados, na ordem que você especificar. Quando um crawler encontra um classificador que corresponde aos dados, o esquema e a string de classificação são usados na definição das tabelas gravadas no seu AWS Glue Data Catalog.

Tópicos

Gravar classificadores grok personalizados

Grok é uma ferramenta usada para analisar dados textuais em um determinado padrão de correspondência. Um padrão grok é um conjunto nomeado de expressões regulares (regex) que são usadas para corresponder dados uma linha por vez. O AWS Glue usa padrões grok usa para inferir o esquema dos seus dados. Quando um padrão grok corresponde aos seus dados, o AWS Glue usa esse padrão para determinar a estrutura dos seus dados e mapeá-los em campos.

O AWS Glue fornece vários padrões integrados. Se preferir, você pode definir o seu próprio padrão. Você pode criar um padrão grok usando padrões integrados e personalizados na definição do seu classificador personalizado. Você pode adaptar um padrão grok para classificar formatos de arquivos de texto personalizados.

> **Note**
>
> Classificadores grok personalizados do AWS Glue usam a biblioteca de serialização `GrokSerDe` para tabelas criadas no AWS Glue Data Catalog. Se você estiver usando o AWS Glue Data Catalog com o Amazon Athena, Amazon EMR ou Redshift Spectrum, verifique a documentação sobre esses serviços para obter informações sobre o suporte do `GrokSerDe`. Atualmente, você pode encontrar problemas ao consultar tabelas criadas com o `GrokSerDe` no Amazon EMR e Redshift Spectrum.

Veja a seguir a sintaxe básica dos componentes de um padrão grok:

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

Os dados que correspondem ao `PATTERN` nomeado são mapeados para a coluna `field-name` no esquema, com um tipo de dados padrão de `string`. Se preferir, o tipo de dados para o campo pode ser convertido em `byte`, `boolean`, `double`, `short`, `int`, `long` ou `float` no esquema resultante.

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

Por exemplo, para converter um campo `num` para um tipo de dados `int`, você pode usar esse padrão:

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

Os padrões podem ser compostos por outros padrões. Por exemplo, você pode ter um padrão para um time stamp SYSLOG que é definido por padrões de mês, dia e horário, como `Feb 1 06:25:43`. Para esses dados, você pode definir o padrão a seguir:

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

> **Note**
>
> Os padrões grok podem processar somente uma linha por vez. Padrões de várias linhas não são compatíveis. Além disso, as quebras de linha dentro de um padrão não são compatíveis.

Valores do classificador personalizado no AWS Glue

Ao definir um classificador grok, você fornece os seguintes valores para que o AWS Glue crie o classificador personalizado.

Nome

Nome do classificador.

Classificação

A string de texto que é escrita para explicar o formato dos dados que serão classificados, por exemplo, `special-logs`.

Padrão grok

O conjunto dos padrões que são aplicados ao armazenamento de dados para determinar se há uma correspondência. Esses são AWS Gluepadrões integrados do e todos os outros padrões personalizados que você definir.

Veja a seguir um exemplo de padrão grok.

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

Quando os dados correspondem a `TIMESTAMP_ISO8601`, uma coluna de esquema `timestamp` é criada. O comportamento é semelhante para os outros padrões nomeados no exemplo.

Padrões personalizados

Padrões personalizados opcionais que você define. Esses padrões são referenciados pelo padrão grok que classifica os seus dados. Você pode referenciar esses padrões personalizados no padrão grok aplicado aos seus dados. Cada padrão de componente personalizado deve estar em uma linha separada. A sintaxe de expressões regulares (regex) é usada para definir o padrão.

Veja a seguir um exemplo de uso dos padrões personalizados:

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

O primeiro padrão nomeado personalizado, `CRAWLERLOGLEVEL`, é uma combinação quando os dados correspondem a uma das strings enumeradas. O segundo padrão personalizado, `MESSAGEPREFIX`, tenta corresponder a uma string de prefixo de mensagem.

O AWS Glue rastreia o horário de criação, o horário da última atualização e a versão do seu classificador.

Padrões integrados do AWS Glue

O AWS Glue fornece vários padrões comuns que você pode usar para criar um classificador personalizado. Você adiciona um padrão nomeado ao `grok pattern` em uma definição de classificador.

A lista a seguir contém uma linha para cada padrão. Em cada linha, o nome do padrão é seguido da sua respectiva definição. A sintaxe de expressões regulares (regex) é usada na definição do padrão.

```
#<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)?)
```

Gravar classificadores XML personalizados

XML define a estrutura de um documento com o uso de tags no arquivo. Com um classificador XML personalizado, você pode especificar o nome de tag usado para definir uma linha.

Valores do classificador personalizado no AWS Glue

Ao definir um classificador XML, você fornece os seguintes valores para que o AWS Glue crie o classificador. O campo de classificação deste classificador é definido como `xml`.

Nome

Nome do classificador.

Etiqueta de linha

O nome da tag XML que define uma linha de tabela no documento XML, sem colchetes angulares < >. O nome precisa estar em conformidade com as regras XML para tags.

> **Note**
>
> O elemento que contém os dados da linha não pode ser um elemento vazio de fechamento automático. Por exemplo, esse elemento vazio não é analisado pelo AWS Glue:
>
> ```
> <row att1=”xx” att2=”yy” />
> ```
>
> Os elementos vazios podem ser escritos da seguinte forma:
>
> ```
> <row att1=”xx” att2=”yy”> </row>
> ```

O AWS Glue rastreia o horário de criação, o horário da última atualização e a versão do seu classificador.

Por exemplo, digamos que você tenha o arquivo XML a seguir. Para criar uma tabela do AWS Glue que contenha apenas colunas para autor e título, crie um classificador no console do AWS Glue com Row tag (Tag de linha) como `AnyCompany`. Em seguida, adicione e execute um crawler que use esse classificador 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>
```

Gravar classificadores JSON personalizados

JSON é um formato de intercâmbio de dados. Ele define estruturas de dados com pares de nome e valor ou uma lista de valores ordenada. Com um classificador JSON personalizado, você pode especificar o caminho JSON de uma estrutura de dados usada para definir o esquema da sua tabela.

Valores do classificador personalizado no AWS Glue

Ao definir um classificador JSON, você fornece os seguintes valores para que o AWS Glue crie esse classificador. O campo de classificação deste classificador é definido como `json`.

Nome

Nome do classificador.

Caminho JSON

Um caminho JSON que aponta para um objeto usado para definir um esquema de tabela. O caminho JSON pode ser escrito na notação de pontos ou de colchetes. Os operadores a seguir são aceitos:

Descrição
Elemento raiz de um objeto JSON. Inicia todas as expressões de caminho
Caracteres curinga. Disponíveis em qualquer lugar, um nome ou número são necessários no caminho JSON.

Descrição
Filho com notação de pontos. Especifica um campo filho em um objeto JSON.
Filho com notação de colchetes. Especifica campo filho em um objeto JSON. Somente um único campo filho pode ser especificado.
Índice de matriz. Especifica o valor de uma matriz por índice.

O AWS Glue rastreia o horário de criação, o horário da última atualização e a versão do seu classificador.

Example Usar um classificador JSON para extrair registros de uma matriz

Suponha que seus dados JSON sejam uma matriz de registros. Por exemplo, as primeiras linhas do seu arquivo podem ter a aparência a seguir:

```
[
  {
    "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"
  }
]
```

Ao executar um crawler utilizando o classificador JSON integrado, o arquivo inteiro é usado para definir o esquema. Como você não especifica um caminho JSON, o crawler trata os dados como um objeto, ou seja, apenas uma matriz. Por exemplo, o esquema pode ser parecido com a seguinte:

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

No entanto, para criar um esquema baseado em cada registro da matriz JSON, crie um classificador JSON personalizado e especifique o caminho JSON como `$[*]`. Quando você especifica esse caminho JSON, o classificador interroga todos os 12 registros na matriz para determinar o esquema. O esquema resultante contém campos separados para cada objeto, semelhante ao exemplo a seguir:

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

Example Usar um classificador JSON para examinar somente partes de um arquivo

Suponha que seus dados JSON sigam o padrão do arquivo JSON de exemplo `s3://awsglue-datasets/examples/us-legislators/all/areas.json` obtido no site http://everypolitician.org/. Exemplos de objetos no arquivo JSON têm a seguinte aparência:

```
{
  "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"
}
```

Ao executar um crawler utilizando o classificador JSON integrado, o arquivo inteiro é usado para criar o esquema. Seu esquema poderá ser semelhante 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
```

No entanto, para criar um esquema usando apenas o objeto "`id`", crie um classificador JSON personalizado e especifique o caminho JSON como `$.id`. Em seguida, o esquema será baseado somente no campo "`id`":

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

As primeiras linhas de dados extraídas com este esquema têm esta aparência:

```
{"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 criar um esquema baseado em um objeto altamente aninhado, como "`identifier`", no arquivo JSON, você pode criar um classificador JSON personalizado e especificar o caminho JSON como `$.identifiers[*].identifier`. Embora o esquema seja parecido com o exemplo anterior, ele é baseado em um objeto diferente no arquivo JSON.

O esquema é semelhante ao seguinte:

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

Listar as primeiras linhas de dados da tabela mostra que o esquema é baseado nos dados no 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 criar uma tabela com base em outro objeto altamente aninhado, como o campo "name" na matriz "other_names" do arquivo JSON, você pode criar um classificador JSON personalizado e especificar o caminho JSON como $.other_names[*].name. Embora o esquema seja parecido com o exemplo anterior, ele é baseado em um objeto diferente no arquivo JSON. O esquema é semelhante ao seguinte:

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

Listar as primeiras linhas de dados na tabela mostra que isto esta baseado nos dados no objeto "name" na 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": "#######"}
```

Gravar classificadores CSV personalizados

Os classificadores CSV personalizados permitem que você especifique tipos de dados para cada coluna no campo de classificador CSV personalizado. É possível especificar o tipo de dados de cada coluna separando-o por uma vírgula. Ao especificar os tipos de dados, é possível substituir os tipos de dados inferidos dos crawlers e garantir que os dados sejam classificados adequadamente.

Você pode definir o SerDe para processamento de CSV no classificador, que será aplicado no catálogo de dados.

Ao criar um classificador personalizado, você também pode reutilizar o classificador para diferentes crawlers.

- Para arquivos csv somente com cabeçalhos (sem dados), esses arquivos serão classificados como UNKNOWN (DESCONHECIDOS), pois não são fornecidas informações suficientes. Se você especificar que o CSV “tem cabeçalhos” na opção Column headings (Cabeçalhos das colunas) e fornecer os tipos de dados, poderemos classificar esses arquivos corretamente.

Você pode usar um classificador CSV personalizado para inferir o esquema de vários tipos de dados CSV. Os atributos personalizados que podem ser fornecidos ao classificador incluem delimitadores, uma opção de SerDe CSV, opções sobre o cabeçalho e se determinadas validações dos dados devem ser executadas.

Valores do classificador personalizado no AWS Glue

Ao definir um classificador CSV, você fornece os seguintes valores para que o AWS Glue crie esse classificador. O campo de classificação deste classificador é definido como `csv`.

Nome do classificador

Nome do classificador.

SerDe CSV

Define o SerDe para processamento de CSV no classificador, que será aplicado no catálogo de dados. As opções são Open SerDe CSV, Lazy Simple SerDe e Nenhum. Você pode especificar o valor Nenhum quando quiser que o crawler faça a detecção.

Delimitador de coluna

Um símbolo personalizado para indicar o que separa cada entrada de coluna na linha. Forneça um caractere Unicode. Se você não conseguir digitar seu delimitador, poderá copiá-lo e colá-lo. Isso funciona para caracteres imprimíveis, incluindo os que não são compatíveis com seu sistema (normalmente exibidos como □).

Símbolo de cotação

Um símbolo personalizado para indicar o que combina o conteúdo em um único valor da coluna. Deve ser diferente do delimitador de coluna. Forneça um caractere Unicode. Se você não

conseguir digitar seu delimitador, poderá copiá-lo e colá-lo. Isso funciona para caracteres imprimíveis, incluindo os que não são compatíveis com seu sistema (normalmente exibidos como □).

Cabeçalhos de coluna

Indica o comportamento de como cabeçalhos de coluna devem ser detectados no arquivo CSV. Se o seu arquivo CSV personalizado tem cabeçalhos de coluna, insira uma lista de cabeçalhos de coluna delimitados por vírgula.

Opções de processamento: permitir arquivos com uma única coluna

Habilita o processamento de arquivos que contêm apenas uma coluna.

Opções de processamento: remover espaço em branco antes de identificar os valores de coluna

Especifica se os valores devem ser removidos antes de identificar o tipo dos valores de coluna.

Tipos de dados personalizados: opcional

Insira o tipo de dados personalizado separado por uma vírgula. Especifica os tipos de dados personalizados no arquivo CSV. O tipo de dados personalizado deve ser um tipo de dados válido. Os tipos de dados válidos são: “BINARY”, “BOOLEAN”, “DATE”, “DECIMAL”, “DOUBLE”, “FLOAT”, “INT”, “LONG”, “SHORT”, “STRING”, “TIMESTAMP”. Tipos de dados inválidos exibirão um erro.

Trabalhar com classificadores no console do AWS Glue

Um classificador determina o esquema dos seus dados. Você pode escrever um classificador personalizado e apontá-lo para o AWS Glue.

Visualizar classificadores

Para ver uma lista com todos os classificadores que você criou, abra o console do AWS Glue em https://console.aws.amazon.com/glue/ e escolha a guia Classifiers (Classificadores).

A lista exibe as seguintes propriedades sobre cada classificador:

- Classifier (Classificador): o nome do classificador. Ao criar um classificador, você precisa fornecer um nome para ele.
- Classification (Classificação): o tipo de classificação das tabelas inferidas pelo classificador.

- Last updated (Última atualização): a hora em que o classificador foi atualizado pela última vez.

Gerenciar classificadores

Na lista Classifiers no console do AWS Glue, você pode adicionar, editar e excluir classificadores. Para ver mais detalhes sobre um classificador, escolha o nome dele na lista. Os detalhes incluem as informações que você definiu quando criou o classificador.

Criar classificadores

Para adicionar um classificador no console do AWS Glue, escolha Add classifier. Ao definir um classificador, você precisa fornecer valores para o seguinte:

- Classifier name (Nome do classificador): forneça um nome exclusivo para o seu classificador.
- Classifier type (Tipo do classificador): o tipo de classificação das tabelas inferidas pelo classificador.
- Last updated (Última atualização): a hora em que o classificador foi atualizado pela última vez.

Nome do classificador

Forneça um nome exclusivo para o seu classificador.

Tipo de classificador

Escolha o tipo de classificador a ser criado.

Dependendo do tipo de classificador escolhido, configure as seguintes propriedades para o classificador:

Grok

- Classificação

  Descreva o formato ou tipo de dados que serão classificados ou forneça um rótulo personalizado.
- Padrão grok

  Isso é usado na análise de dados em um esquema estruturado. O padrão grok é composto por padrões nomeados que descrevem o formato do seu armazenamento de dados. Você

escreve esse padrão grok usando os padrões integrados nomeados fornecidos pelo AWS Glue e os padrões personalizados incluídos no campo Custom patterns. Embora os resultados do depurador grok possam não corresponder exatamente aos resultados do AWS Glue, sugerimos que você teste seu padrão usando alguns dados de amostra com um depurador grok. Você pode encontrar os depuradores grok na web. Os padrões integrados nomeados fornecidos pelo AWS Glue são geralmente compatíveis com os padrões grok disponíveis na web.

Crie o seu padrão grok adicionando padrões nomeados de forma interativa e verifique seus resultados em um depurador. Ao fazer isso, você garante que seus dados poderão ser analisados quando o crawler do AWS Glue executar seu padrão grok.

- Padrões personalizados

  Para os classificadores grok, estes são elementos fundamentais para o Grok pattern que você escreveu. Quando os padrões integrados não conseguirem analisar seus dados, você precisará escrever um padrão personalizado. Esses padrões personalizados são definidos neste campo e referenciados em Grok pattern. Cada padrão personalizado é definido em uma linha separada. Assim como os padrões integrados, ele consiste em uma definição de padrão nomeado que usa a sintaxe de expressões regulares (regex).

  Por exemplo, o nome `MESSAGEPREFIX` a seguir apresenta uma definição de expressão regular a ser aplicada aos seus dados para determinar se eles seguem o padrão.

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

XML

- Etiqueta de linha

  Para classificadores XML, este é o nome da tag XML que define uma linha de tabela no documento XML. Digite o nome sem colchetes angulares < >. O nome precisa estar em conformidade com as regras XML para tags.

  Para ter mais informações, consulte Gravar classificadores XML personalizados.

JSON

- Caminho JSON

Para os classificadores JSON, este é o caminho JSON para o objeto, a matriz ou o valor que definem uma linha da tabela que está sendo criada. Digite o nome na sintaxe JSON de ponto ou colchete usando os operadores do AWS Glue compatíveis.

Para obter mais informações, consulte a lista de operadores em Gravar classificadores JSON personalizados.

CSV

- Delimitador de coluna

  Um único caractere ou símbolo para indicar o que separa cada entrada de coluna na linha. Escolha o delimitador da lista, ou escolha `Other` para inserir um delimitador personalizado.

- Símbolo de cotação

  Um único caractere ou símbolo para indicar o que combina o conteúdo em um único valor de coluna. Deve ser diferente do delimitador de coluna. Escolha o símbolo de aspas na lista, ou escolha `Other` para inserir um caractere de aspas personalizado.

- Cabeçalhos de coluna

  Indica o comportamento de como cabeçalhos de coluna devem ser detectados no arquivo CSV. Você pode escolher `Has headings`, `No headings` ou `Detect headings`. Se o seu arquivo CSV personalizado tem cabeçalhos de coluna, insira uma lista de cabeçalhos de coluna delimitados por vírgula.

- Permitir arquivos com uma única coluna

  Para serem classificados como CSV, os dados devem ter pelo menos duas colunas e duas linhas de dados. Use esta opção para permitir o processamento de arquivos que contenham apenas uma coluna.

- Remover espaço em branco antes de identificar os valores de coluna

  Esta opção especifica se os valores devem ser removidos antes de identificar o tipo dos valores de coluna.

- Tipo de dados personalizado

(Opcional) Insira tipos de dados personalizados em uma lista delimitada por vírgulas. Os tipos de dados válidos são: “BINARY”, “BOOLEAN”, “DATE”, “DECIMAL”, “DOUBLE”, “FLOAT”, “INT”, “LONG”, “SHORT”, “STRING”, “TIMESTAMP”.

- Serde CSV

  (Opcional): um SerDe para processamento de CSV no classificador, que será aplicado no catálogo de dados. Escolha entre `Open CSV SerDe`, `Lazy Simple SerDe`, ou `None`. Você pode especificar o valor `None` quando quiser que o crawler faça a detecção.

Para ter mais informações, consulte Gravar classificadores personalizados.

Programar um crawler do AWS Glue

Você pode executar um crawler do AWS Glue sob demanda ou por meio de uma programação regular. As programações do crawler podem ser expressas no formato cron. Para obter mais informações, consulte cron na Wikipédia.

Ao criar um crawler com base em uma programação, você pode especificar certas restrições (por exemplo: a frequência com que o crawler será executado, em quais dias da semana ele será executado e a que horas). Essas restrições são feitas com base no cron. Ao configurar a programação de um crawler, você precisa considerar os recursos e as limitações do cron. Por exemplo, se você optar por executar seu crawler no dia 31 de cada mês, lembre-se de que alguns meses não têm 31 dias.

Os crawls para cada crawler são válidos apenas por até 12 meses

Para obter mais informações sobre como usar o cron para programar trabalhos e crawlers, consulte Programações baseadas em hora para trabalhos e crawlers.

Visualizar resultados e detalhes do crawler

Depois que o crawler é executado com êxito, ele cria definições de tabela no Data Catalog. Escolha Tables (Tabelas) no painel de navegação para ver as tabelas criadas pelo seu crawler no banco de dados que você especificou.

Você pode exibir informações relacionadas ao próprio crawler da seguinte forma:

- A página Crawlers no console do AWS Glue exibe as seguintes propriedades para um crawler:

Propriedade	Descrição
Nome	Ao criar um crawler, você precisa atribuir a ele um nome único.
Status	Um crawler pode estar pronto, sendo iniciado, interrompido, programado ou com a programação em pausa. Um crawler em execução progride desde o início até a interrupção. Você pode retomar ou pausar uma programação vinculada a um crawler.
Schedule (Programação)	Você pode optar por executar seu crawler sob demanda ou escolher a frequência desejada usando uma programação. Para obter mais informações sobre como programar um crawler, consulte Programar um crawler.
Last run (Última execução)	A data e a hora da última vez em que o crawler foi executado.
Log (Log)	Links para todos os logs disponíveis referente s à última execução do crawler.
Tables changes from last run (Alterações de tabelas desde a última execução)	A quantidade das tabelas no AWS Glue Data Catalog que foram atualizadas pela última execução do crawler.

- Para exibir o histórico de um crawler, escolha Crawlers no painel de navegação para ver os crawlers que você criou. Escolha um crawler na lista de crawlers disponíveis. Você pode ver as propriedades do crawler e o histórico do crawler na guia Crawler runs (Execuções de crawler).

  A guia Crawler runs (Execuções de crawler) exibe informações sobre cada vez que o crawler foi executado, incluindo Start time (UTC) (Horário inicial [UTC]), End time (UTC) (Horário final [UTC]), Duration (Duração), Status (Status), DPU hours (Horas de DPU) e Table changes (Alterações de tabela).

A guia de execuções do crawler exibe apenas os crawls que ocorreram desde a data de lançamento do atributo de histórico do crawler e só retém os crawls por 12 meses. Crawls mais antigos não serão retornados.

- Para ver informações adicionais, escolha uma guia na página de detalhes do crawler. Cada guia exibirá informações relacionadas ao crawler.
  - Schedule (Cronograma): todas as programações criadas para o crawler estarão visíveis aqui.
  - Data sources (Fontes de dados): todas as fontes de dados rastreadas pelo crawler estarão visíveis aqui.
  - Classifiers (Classificadores): todos os classificadores atribuídos ao crawler estarão visíveis aqui.
  - Tags (Tags): todas as tags criadas e atribuídas a um recurso do AWS estarão visíveis aqui.

Parâmetros definidos nas tabelas do Data Catalog pelo crawler

Essas propriedades da tabela são definidas pelos crawlers do AWS Glue. Esperamos que os usuários consumam as propriedades `classification` e `compressionType`. Outras propriedades, incluindo estimativas de tamanho de tabela, são usadas para cálculos internos e não garantimos sua precisão ou aplicabilidade aos casos de uso do cliente. Alterar esses parâmetros pode alterar o comportamento do crawler. Não oferecemos suporte a esse fluxo de trabalho.

Propriedade da chave	Valor da propriedade
`UPDATED_BY_CRAWLER`	Nome do crawler que está executando a atualização.
`connectionName`	O nome da conexão no Data Catalog para o crawler usado para conexão com o armazenamento de dados.
`recordCount`	Estime a contagem de registros na tabela com base nos tamanhos e cabeçalhos dos arquivos.
`skip.header.line.count`	Linhas ignoradas para pular o cabeçalho. Definido em tabelas classificadas como CSV.
`CrawlerSchemaSerializerVersion`	Para uso interno

Propriedade da chave	Valor da propriedade
`classification`	Formato dos dados, inferido pelo crawler. Para obter mais informaçõ es sobre os formatos de dados aceitos pelos crawlers do AWS Glue, consulte the section called “Classificadores integrados no AWS Glue”.
`CrawlerSc hemaDeser ializerVersion`	Para uso interno
`sizeKey`	Tamanho combinado dos arquivos na tabela com crawling.
`averageRe cordSize`	O tamanho médio da linha na tabela, em bytes.
`compressionType`	Tipo de compressão usada nos dados da tabela. Para obter mais informações sobre os tipos de compressão aceitos pelos crawlers do AWS Glue, consulte the section called “Classificadores integrados no AWS Glue”.
`typeOfData`	`file`, `table` ou `view`.
`objectCount`	Número de objetos no caminho do Amazon S3 para a tabela.

Essas propriedades adicionais da tabela são definidas por crawlers do AWS Glue para armazenamentos de dados do Snowflake.

Propriedade da chave	Valor da propriedade
`aws:RawTa bleLastAltered`	Registra o timestamp da última alteração da tabela do Snowflake.
`ViewOrigi nalText`	Visualizar a instrução SQL.
`ViewExpan dedText`	Visualizar a instrução SQL codificada no formato Base64.

Propriedade da chave	Valor da propriedade
`ExternalTable:S3Location`	O local do Amazon S3 da tabela externa do Snowflake.
`ExternalTable:FileFormat`	O formato do arquivo do Amazon S3 da tabela externa do Snowflake.

Essas propriedades adicionais da tabela são definidas por crawlers do AWS Glue para armazenamentos de dados do tipo JDBC, como Amazon Redshift, Microsoft SQL Server, MySQL, PostgreSQL e Oracle.

Propriedade da chave	Valor da propriedade
`aws:RawType`	Quando um crawler armazena os dados no Dta Catalog, ele converte os tipos de dados em tipos compatíveis com o Hive, o que, muitas vezes, faz com que as informações sobre o tipo de dados nativo sejam perdidas. O crawler gera o parâmetro `aws:RawType` para fornecer o tipo de dados de nível nativo.
`aws:RawColumnComment`	Se um comentário estiver associado a uma coluna no banco de dados, o crawler gera o comentário correspondente na tabela do catálogo. A string de comentários é truncada em 255 bytes. O Microsoft SQL Server não oferece suporte para comentários.
`aws:RawTableComment`	Se um comentário estiver associado a uma tabela no banco de dados, o crawler gera o comentário correspondente na tabela do catálogo. A string de comentários é truncada em 255 bytes. O Microsoft SQL Server não oferece suporte para comentários.

Personalizar o comportamento do Crawler

Quando um crawler é executado, ele pode encontrar alterações em seu datastore que resultam em um esquema ou partição diferente de um rastreamento anterior. Você pode usar o AWS Management Console ou a API do AWS Glue para configurar como o seu crawler processa determinados tipos de alterações.

Tópicos

Console

Quando você define um crawler usando o console do AWS Glue, tem várias opções para configurar o comportamento desse crawler. Para obter mais informações sobre como usar o console do AWS Glue para adicionar um crawler, consulte Configurar um crawler.

Quando um crawler é executado em um datastore previamente rastreado, ele pode descobrir que um esquema foi alterado ou que alguns objetos no datastore foram excluídos. O crawler registra as alterações em um esquema. Dependendo do tipo de origem do crawler, novas tabelas e partições podem ser criadas independentemente da política de alteração do esquema.

Para especificar o que o crawler faz quando encontra alterações no esquema, você pode escolher uma das seguintes ações no console:

- Update the table definition in the Data Catalog (Atualizar a definição da tabela no Data Catalog): adicione novas colunas, remova colunas ausentes e modifique as definições de colunas existentes no AWS Glue Data Catalog. Remova todos os metadados não definidos pelo crawler. Essa é a configuração padrão.

- Add new columns only (Adicionar somente novas colunas): para tabelas que são mapeadas para um datastore do Amazon S3, inclua novas colunas conforme elas são descobertas, mas não remova ou altere o tipo de colunas existentes no Data Catalog. Escolha essa opção quando as colunas atuais no Data Catalog estiverem corretas e você não quiser que o crawler remova ou altere o tipo das colunas existentes. Se um atributo de tabela fundamental do Amazon S3 for alterado, como uma classificação, tipo de compactação ou um delimitador de CSV, marque a tabela como defasada. Mantenha os formatos de entrada e de saída existentes no Data Catalog. Atualize os parâmetros de SerDe somente se o parâmetro for definido pelo crawler. For all other data stores, modify existing column definitions (Para todos os outros armazenamentos de dados, modifique as definições de coluna existentes).
- Ignore the change and don't update the table in the Data Catalog (Ignorar a alteração e não atualizar a tabela no Data Catalog): somente novas tabelas e partições são criadas.

  Essa é a configuração padrão para crawls incrementais.

Um crawler também pode descobrir partições novas ou alteradas. Por padrão, novas partições são adicionadas e as partições existentes são atualizadas se houver mudanças. Além disso, você pode definir uma opção de configuração de crawler como Update all new and existing partitions with metadata from the table (Atualizar todas as partições novas e existentes com metadados da tabela) no console do AWS Glue. Quando essa opção é definida, as partições herdam da tabela pai as propriedades dos metadados, como classificação, formato de entrada, formato de saída, informações de SerDe e esquema. Todas as alterações nessas propriedades em uma tabela são propagadas para suas respectivas partições. Quando essa opção de configuração é definida em um crawler existente, as partições existentes são atualizadas para corresponder às propriedades da tabela pai na próxima vez que o crawler for executado.

Para especificar o que o crawler faz quando encontra um objeto excluído no datastore, escolha uma das seguintes ações:

- Excluir tabelas e partições do Data Catalog
- Ignorar a alteração e não atualizar a tabela no Data Catalog

  Essa é a configuração padrão para crawls incrementais.
- Mark the table as deprecated in the Data Catalog (Marcar a tabela como defasada no Data Catalog): essa é a configuração padrão.

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

Quando você define um crawler usando a API do AWS Glue, é possível escolher entre vários campos para configurar o crawler. A `SchemaChangePolicy` na API do crawler determina o que o crawler faz quando descobre um esquema alterado ou um objeto excluído. O crawler registra alterações do esquema à medida que ele é executado.

Exemplo de código python mostrando as opções de configuração do crawler

```
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
```

Quando um crawler é executado, novas tabelas e partições sempre são criadas independentemente da política de alteração de esquema. Você pode escolher uma das seguintes ações no campo `UpdateBehavior` na estrutura `SchemaChangePolicy` para determinar o que o crawler faz quando encontra um esquema de tabela alterado:

- `UPDATE_IN_DATABASE`: atualizar a tabela no AWS Glue Data Catalog. Adicione novas colunas, remova as colunas ausentes e modifique as definições das colunas existentes. Remova todos os metadados não definidos pelo crawler.
- `LOG`: ignorar as alterações e não atualizar a tabela no Data Catalog.

  Essa é a configuração padrão para crawls incrementais.

Você também pode substituir a estrutura `SchemaChangePolicy` usando um objeto JSON fornecido no campo `Configuration` da API do crawler. Esse objeto JSON pode conter um par chave-valor para definir a política para não atualizar as colunas existentes e adicionar apenas novas colunas. Por exemplo, forneça o seguinte objeto JSON como uma string:

```
{
```

```
   "Version": 1.0,
   "CrawlerOutput": {
      "Tables": { "AddOrUpdateBehavior": "MergeNewColumns" }
   }
}
```

Essa opção corresponde à opção Add new columns only (Adicionar somente novas colunas) no console do AWS Glue. Ela substitui a estrutura `SchemaChangePolicy` para tabelas que resultam somente do crawling de armazenamentos de dados do Amazon S3. Escolha essa opção se quiser manter os metadados existentes no Data Catalog (a fonte confiável). As novas colunas são adicionadas conforme são encontradas, incluindo tipos de dados aninhados. Mas as colunas existentes não são removidas e o tipo não é alterado. Se um atributo de tabela do Amazon S3 for alterado significativamente, marque a tabela como defasada e registre em log um aviso de que um atributo incompatível precisa ser resolvido. Essa opção não se aplica ao crawler incremental.

Quando um crawler é executado em um datastore previamente rastreado, ele pode descobrir partições novas ou alteradas. Por padrão, novas partições são adicionadas e as partições existentes são atualizadas se houver mudanças. Além disso, você pode definir uma opção de configuração do crawler como `InheritFromTable`, que corresponde à opção Update all new and existing partitions with metadata from the table (Atualizar todas as partições novas e existentes com metadados da tabela) no console do AWS Glue. Quando essa opção é definida, as partições herdam as propriedades dos metadados de sua tabela principal, como classificação, formato de entrada, formato de saída, informações de SerDe e esquema. Todas propriedades alteradas na tabela principal são propagadas para suas respectivas partições.

Quando essa opção de configuração é definida em um crawler existente, as partições existentes são atualizadas para corresponder às propriedades da tabela pai na próxima vez que o crawler for executado. Esse comportamento é definido no campo `Configuration` da API do crawler. Por exemplo, forneça o seguinte objeto JSON como uma string:

```
{
   "Version": 1.0,
   "CrawlerOutput": {
      "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" }
   }
}
```

O campo `Configuration` da API do crawler pode definir várias opções de configuração. Por exemplo, para configurar a saída do crawler para partições e tabelas, você pode fornecer uma representação de string do seguinte objeto JSON:

```
{
    "Version": 1.0,
    "CrawlerOutput": {
       "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" },
       "Tables": {"AddOrUpdateBehavior": "MergeNewColumns" }
    }
}
```

Você pode escolher uma das seguintes ações para determinar o que o crawler faz quando encontra um objeto excluído no datastore. O campo `DeleteBehavior` na estrutura `SchemaChangePolicy` na API do crawler define o comportamento do crawler quando ele descobre um objeto excluído.

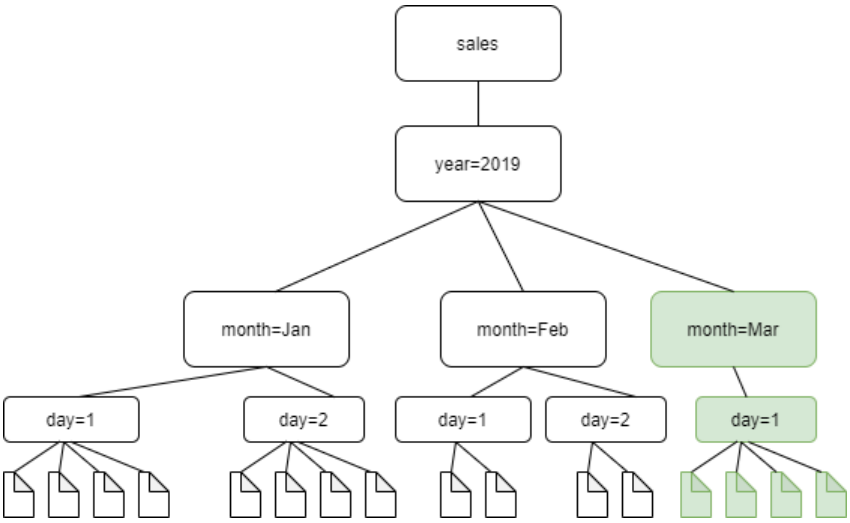
- `DELETE_FROM_DATABASE`: excluir tabelas e partições do Data Catalog.
- `LOG`: ignorar a alteração. Não atualizar o Data Catalog. Em vez disso, grave uma mensagem de log.
- `DEPRECATE_IN_DATABASE`: marcar a tabela como defasada no Data Catalog. Essa é a configuração padrão.

Crawls incrementais para adicionar novas partições

O crawler fornece uma opção para adicionar novas partições, resultando em crawls mais rápidos para conjuntos de dados incrementais com um esquema de tabela estável. O caso de uso típico é para crawlers programados, em que durante cada crawl, novas partições são adicionadas. Quando essa opção é ativada, ela executa primeiro um crawl completo no conjunto de dados de destino para permitir que o crawler registre o esquema inicial e a estrutura de partição. Durante um novo crawl, novas partições serão adicionadas às tabelas existentes somente quando os esquemas forem compatíveis. Nenhuma alteração no esquema é feita e nenhuma tabela nova será adicionada ao Catálogo de Dados após a primeira execução do crawl.

É possível usar essa opção ao configurar uma fonte de dados do Amazon S3. Você pode definir o **`RecrawlPolicy`** with **`RecrawlBehavior`** como “Crawl_New_Folders” na **`CreateCrawler`** API ou o crawler subsequente é executado como Crawl novas subpastas somente no console.

Continuando com o exemplo em the section called “Como um crawler determina quando criar partições?”, o diagrama a seguir mostra que os arquivos do mês de março foram adicionados.



Se você definir a opção `RecrawlBehavior` como “Crawl_New_Folders”, somente a nova pasta `month=Mar` e rastreada.

Notas e restrições

Quando essa opção está ativada, não é possível alterar os armazenamentos de dados de destino do Amazon S3 ao editar o crawler. Essa opção afeta determinadas definições de configuração do crawler. Quando ativada, ela força o comportamento de atualização e de exclusão do crawler para `LOG`. Isto significa que:

- Se descobrir objetos nos quais os esquemas não são compatíveis, o crawler não adicionará os objetos no Catálogo de Dados e adicionará esse detalhe como um log no CloudWatch Logs.
- Ele não atualizará objetos excluídos no Catálogo de dados.

Para ter mais informações, consulte the section called “Personalizar o comportamento do Crawler”.

Definir a opção de configuração do crawler de índice de partição

O catálogo de dados é compatível com índices de partição para fornecer uma pesquisa eficiente de partições específicas. Para obter mais informações, consulte Working with partition indexes in AWS Glue. O crawler do AWS Glue cria índices de partição para destinos do Amazon S3 e do Delta Lake por padrão.

Quando você define um crawler, a opção Criar índices de partição automaticamente é habilitada por padrão em Opções avançadas na página Definir saída e agendamento.

Para desabilitar essa opção, você pode desmarcar a caixa de seleção Criar índices de partição automaticamente no console. Também é possível desabilitar essa opção usando a API do crawler, defina o `CreatePartitionIndex` em `Configuration`. O valor padrão é true.

Notas de uso para índices de partição

- As tabelas criadas pelo crawler não têm a variável `partition_filtering.enabled` por padrão. Para obter mais informações, consulte AWS Glue partition indexing and filtering.
- A criação de índices de partição para partições criptografadas não é compatível.

Acelerar crawls usando notificações de eventos do Amazon S3

Em vez de listar os objetos de um destino do Amazon S3 ou do Data Catalog, você pode configurar o crawler para usar eventos do Amazon S3 para localizar quaisquer alterações. Esse recurso melhora o tempo de recrawl usando eventos do Amazon S3 para identificar as alterações entre dois crawls listando todos os arquivos da subpasta que disparou o evento em vez de listar o destino completo do Amazon S3 ou do Data Catalog.

O primeiro crawl lista todos os objetos do Amazon S3 do destino. Após o primeiro crawling bem-sucedido, você pode optar pelo recrawling manualmente ou em um cronograma definido. O crawler listará apenas os objetos desses eventos em vez de listar todos os objetos.

As vantagens de migrar para um crawler baseado em eventos do Amazon S3 são:

- Uma recuperação mais rápida, pois a listagem de todos os objetos do destino não é necessária, em vez disso, a listagem de pastas específicas é feita onde os objetos são adicionados ou excluídos.
- Uma redução no custo geral de crawl à medida que a listagem de pastas específicas é feita onde os objetos são adicionados ou excluídos.

O crawl de eventos do Amazon S3 é executado consumindo eventos do Amazon S3 da fila SQS com base na programação do crawler. Não haverá custo se não houver eventos na fila. Os eventos do Amazon S3 podem ser configurados para ir diretamente para a fila SQS ou nos casos em que vários consumidores precisam do mesmo evento, uma combinação de SNS e SQS. Para ter mais informações, consulte the section called “Configurar sua conta para notificações de eventos do Amazon S3”.

Depois de criar e configurar o crawler no modo de evento, o primeiro crawl é executado no modo de listagem fazendo uma listagem completa do destino do Amazon S3 ou do Data Catalog. O log a seguir confirma a operação do crawl consumindo eventos do Amazon S3 após o primeiro crawl bem-sucedido: "The crawl is running by consuming Amazon S3 events." (O crawl está sendo executado consumindo eventos do Amazon S3).

Depois de criar o crawl de eventos do Amazon S3 e atualizar as propriedades do crawler que podem afetar o crawl, o crawl opera no modo de lista e o seguinte log é adicionado: "Crawl is not running in S3 event mode" (O crawl não está sendo executado no modo de evento S3).

> **Note**
>
> O número máximo de mensagens a serem consumidas é 10.000 mensagens por operação de crawl.

Destino: Catalog

Quando o destino é o Data Catalog, o crawler atualiza as tabelas existentes no Data Catalog com as alterações (por exemplo, partições extras em uma tabela).

Tópicos

- Configurar sua conta para notificações de eventos do Amazon S3
- Usar criptografia com o crawler de eventos do Amazon S3

Configurar sua conta para notificações de eventos do Amazon S3

Esta seção descreve como configurar sua conta para notificações de eventos do Amazon S3 e fornece instruções para fazer isso usando um script ou o console do AWS Glue.

Pré-requisitos

Complete as tarefas de configuração a seguir. Observe que os valores entre parênteses fazem referência às definições configuráveis do script.

1. Crie um bucket do Amazon S3 (`s3_bucket_name`).
2. Identificar um destino de crawler (`folder_name`, como "test1"), que é um caminho no bucket identificado.
3. Preparar um nome de crawler (`crawler_name`)

4. Preparar um nome de tópico do SNS (`sns_topic_name`) que poderia ser o mesmo que o nome do crawler.
5. Preparar a região da AWS em que o crawler deve ser executado e o bucket do S3 existe (`region`).
6. Opcionalmente, prepare um endereço de email se o email for usado para obter os eventos do Amazon S3 (`subscribing_email`).

Você também pode usar a pilha do CloudFormation para criar seus recursos. Execute as etapas a seguir:

1. Iniciar a pilha do CloudFormation no Leste dos EUA (Norte da Virgínia):
2. Em Parâmetros, insira um nome para o bucket do Amazon S3 (inclua o número da sua conta).
3. Selecione `I acknowledge that AWS CloudFormation might create IAM resources with custom names`.
4. Selecione `Create stack`.

Limitações:

- O crawler só é compatível com um único destino, tanto para destinos do Amazon S3 quanto do Data Catalog.
- Não há suporte para SQS na VPC privada.
- Não há suporte para amostragem do Amazon S3.
- O destino do crawler deve ser uma pasta para um destino do Amazon S3 ou uma ou mais tabelas do AWS Glue Data Catalog para um destino do Data Catalog.
- Não há suporte para o curinga de caminho ‘everything’ (tudo): s3: //%
- Para um destino do Data Catalog, todas as tabelas de catálogo devem apontar para o mesmo bucket do Amazon S3 para o modo de evento do Amazon S3.
- Para um destino do Data Catalog, uma tabela de catálogo não deve apontar para uma localização no Amazon S3 no formato Delta Lake (contendo pastas _symlink ou conferindo o `InputFormat` da tabela do catálogo).

Para usar o crawler baseado em eventos do Amazon S3, você deve habilitar a notificação de eventos no bucket do S3 com eventos filtrados do prefixo que é o mesmo que o destino do S3 e armazenado no SQS. Você pode configurar o SQS e a notificação de eventos por meio do console seguindo as

etapas em Passo a passo: configuração de um bucket para notificações ou usando othe section called “Script para gerar SQS e configurar eventos do Amazon S3 com base no destino”.

Política de SQS

Adicione a seguinte política de SQS que é necessária para ser anexada à função usada pelo crawler.

```
{
    "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 gerar SQS e configurar eventos do Amazon S3 com base no destino

Depois de garantir que os pré-requisitos sejam atendidos, você poderá executar o script Python a seguir para criar o SQS. Substitua as definições configuráveis pelos nomes preparados a partir dos pré-requisitos.

> **Note**
>
> Depois de executar o script, faça login no console do SQS para encontrar o ARN do SQS criado.

O Amazon SQS define um tempo limite de visibilidade, um período durante o qual o Amazon SQS impede que outros consumidores recebam e processem a mensagem. Defina o tempo limite de visibilidade aproximadamente igual ao runtime do crawl.

```
#!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.')
```

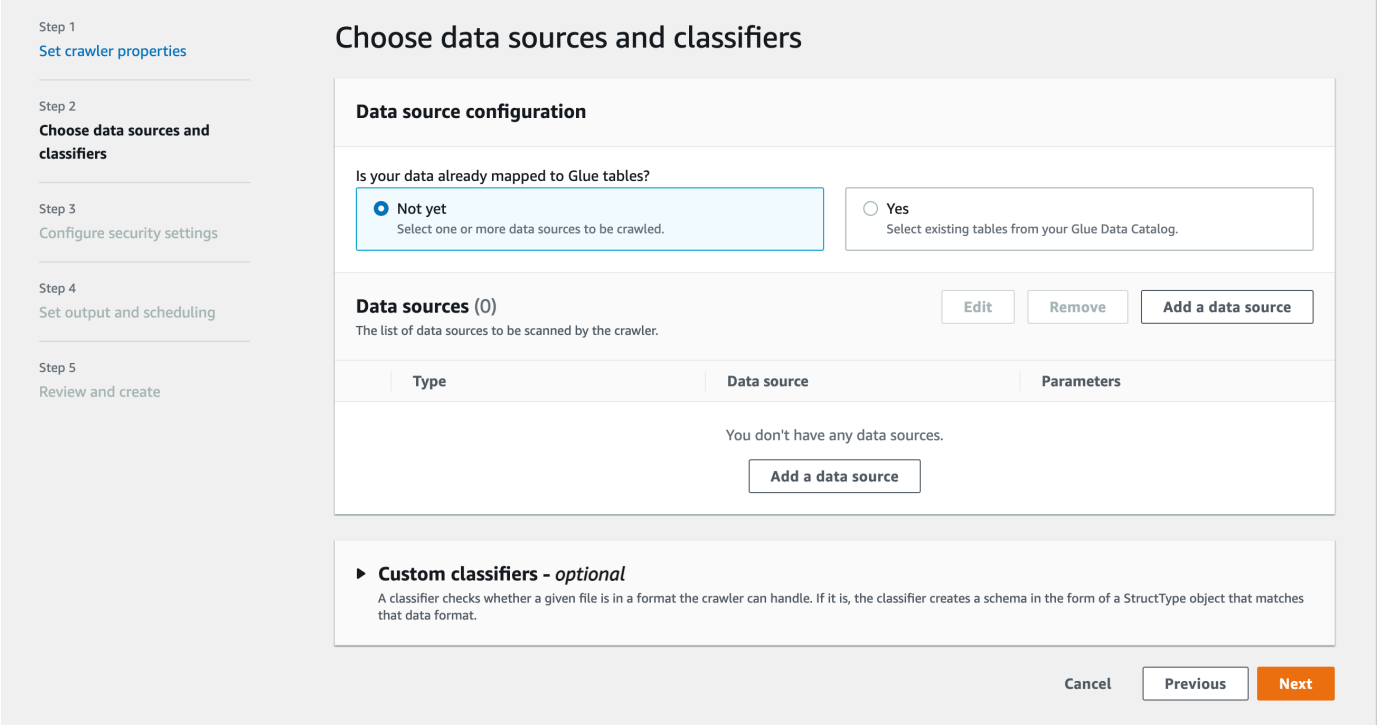
```
            else:
                print('Failed to add SNS topic policy.')
        else:
            print('Failed to create SNS topic.')
```

Configurar um crawler para fornecer notificações de eventos do Amazon S3 usando o console (destino do Amazon S3)

Para configurar um crawler para fornecer notificações de eventos do Amazon S3 usando o console do AWS Glue para um destino do Amazon S3:

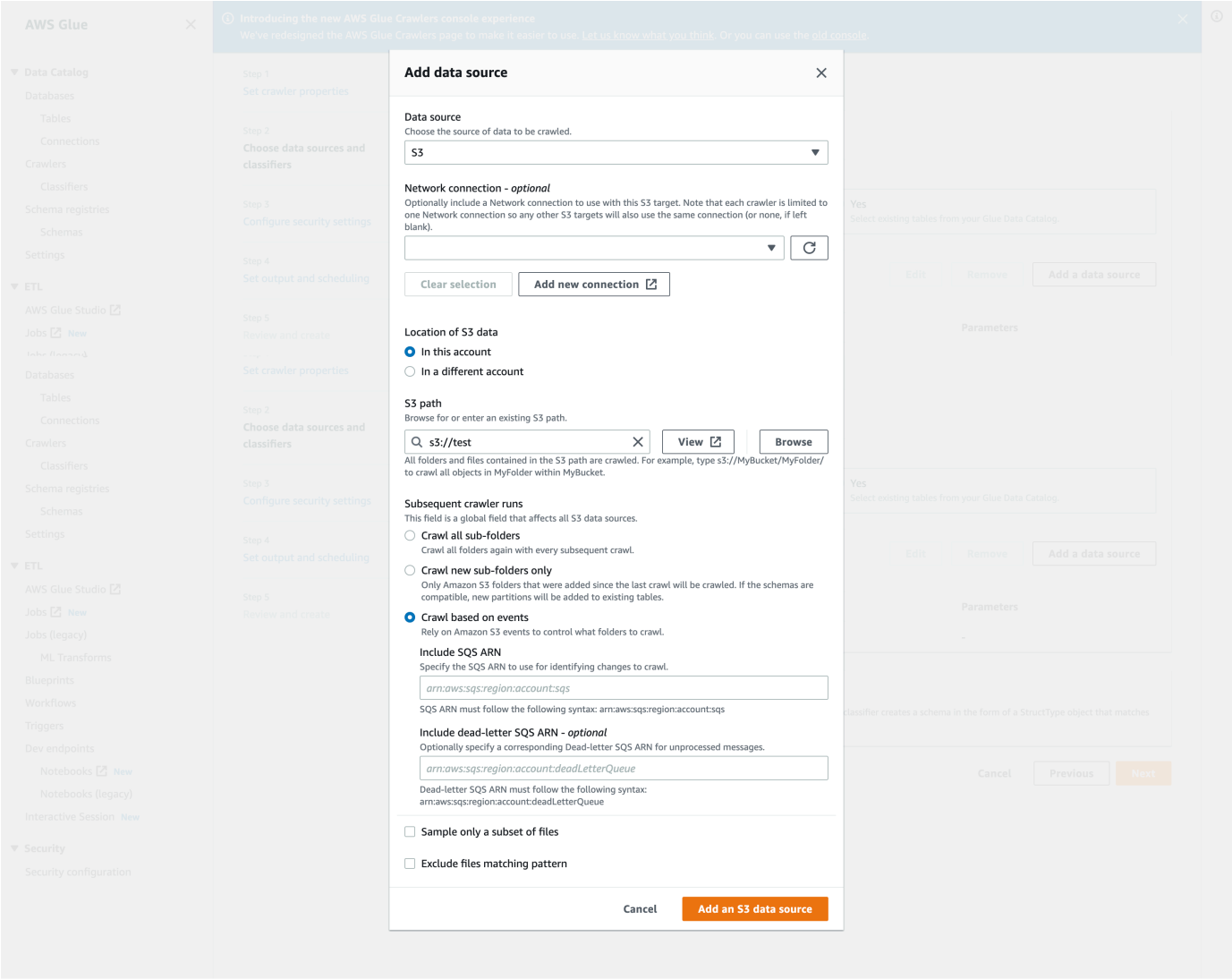
1. Defina as propriedades do crawler. Para obter mais informações, consulte Definir opções de configuração do crawler no console do AWS Glue.
2. Na seção Configuração da fonte de dados, é perguntado a você se Os dados já estão mapeados para tabelas do AWS Glue?

   A opção Not yet (Ainda não) já estará selecionada por padrão. Isso ocorre porque você está usando uma fonte de dados do Amazon S3 e os dados ainda não estão mapeados para as tabelas do AWS Glue.
3. Na seção Data sources (Fontes de dados), escolha Add a data source (Adicionar uma fonte de dados).



4. No modal Add data source (Adicionar origem dos dados), configure a fonte de dados do Amazon S3:

   - Data source (Fonte de dados): o Amazon S3 é selecionado por padrão.
   - Network connection (Conexão de rede) (opcional): escolha Add new connection (Adicionar nova conexão).
   - Location of Amazon S3 data (Local de dados do Amazon S3): a opção In this account (Nesta conta) será selecionado por padrão.
   - Amazon S3 path (Caminho do Amazon S3): especifique o caminho do Amazon S3 no qual pastas e arquivos são rastreados.
   - Subsequent crawler runs (Execuções subsequentes do crawler): escolha Crawl based on events (Rastreamento baseado em eventos) para usar notificações de eventos do Amazon S3 para seu crawler.
   - Include SQS ARN (Incluir ARN de SQS): especifique os parâmetros do armazenamento de dados, incluindo um ARN válido do SQS. (Por exemplo, `arn:aws:sqs:region:account:sqs`).
   - Include dead-letter SQS ARN (Incluir ARN de mensagens não entregues do SQS) (opcional): especifique um ARN válido de mensagens não entregues do SQS na Amazon. (Por exemplo, `arn:aws:sqs:region:account:deadLetterQueue`).
   - Escolha Add an Amazon S3 data source (Adicionar uma fonte de dados do Amazon S3).



Configurar um crawler para notificações de eventos do Amazon S3 usando o console do AWS CLI

Veja a seguir um exemplo de chamada da AWS CLI do Amazon S3 para criar filas SQS e configurar notificações de eventos no bucket de destino do 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"
                    }
                ]
            }
        }
    }
]
}
```

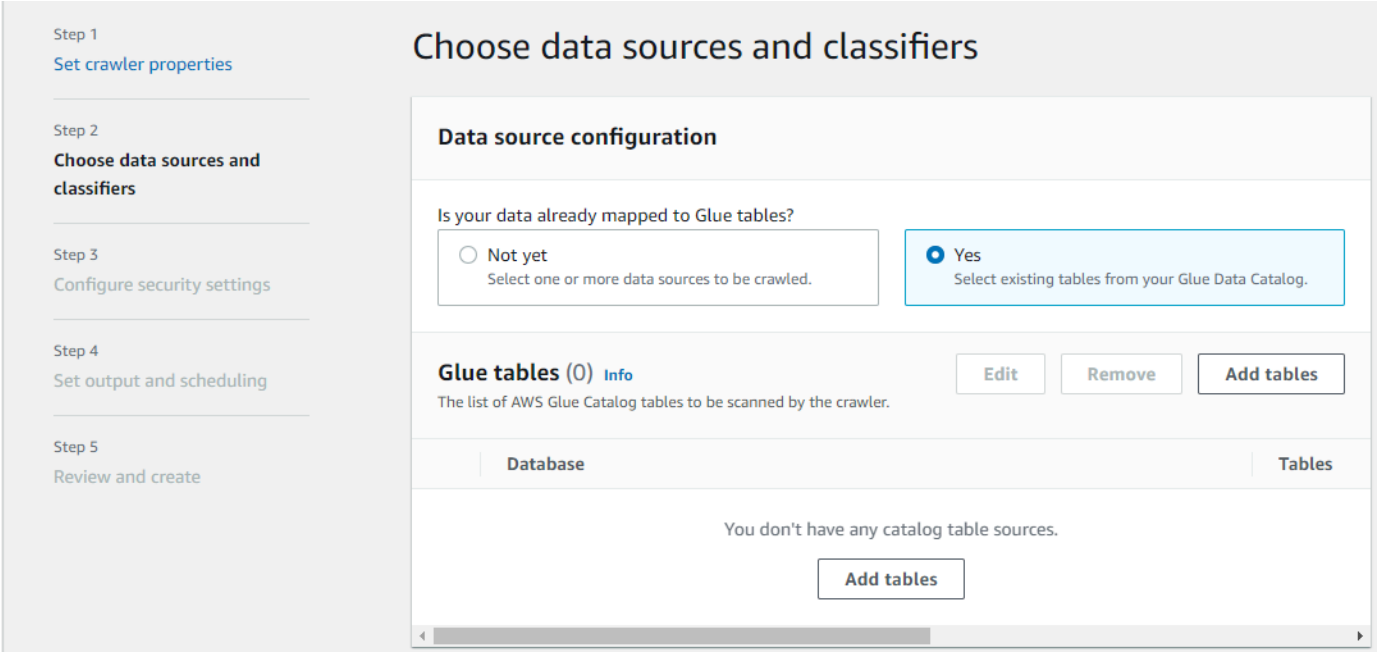
Create Crawler:

Configurar um crawler para fornecer notificações de eventos do Amazon S3 usando o console (destino do Data Catalog)

Quando você tiver um destino de catálogo, configure um crawler para fornecer notificações de eventos do Amazon S3 usando o console do AWS Glue:

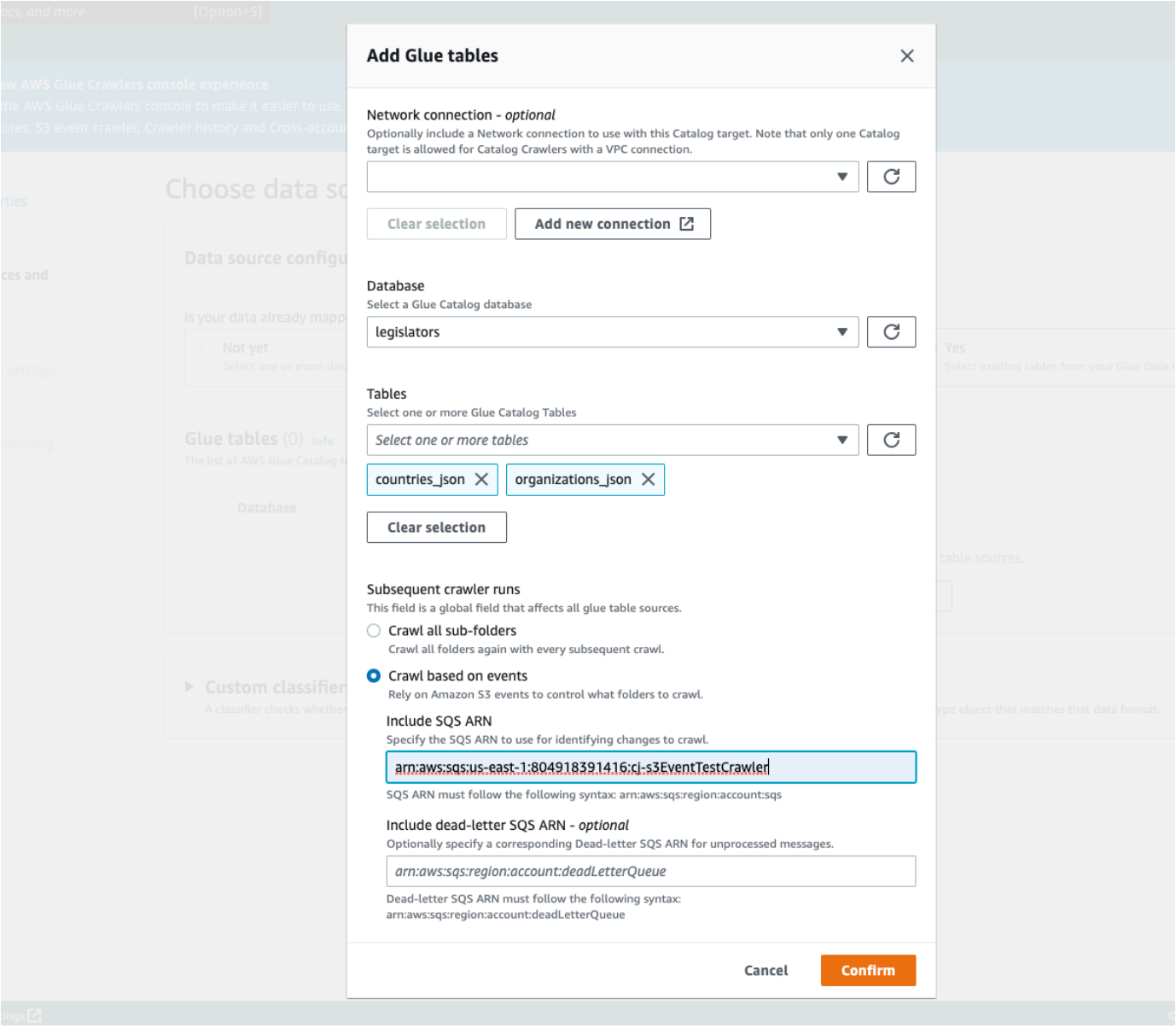
1. Defina as propriedades do crawler. Para obter mais informações, consulte Definir opções de configuração do crawler no console do AWS Glue.
2. Na seção Configuração da fonte de dados, é perguntado a você se Os dados já estão mapeados para tabelas do AWS Glue?

   Selecione Yes (Sim) para selecionar as tabelas existentes do Data Catalog como fonte de dados.
3. Na seção Glue tables (Tabelas do Glue), escolha Add tables (Adicionar tabelas).



4. No modal Add table (Adicionar tabela), configure o banco de dados e as tabelas:

   - Network connection (Conexão de rede) (opcional): escolha Add new connection (Adicionar nova conexão).
   - Database (Banco de dados): selecione um banco de dados no Data Catalog.
   - Tables (Tabelas): selecione uma ou mais tabelas desse banco de dados no Data Catalog.
   - Subsequent crawler runs (Execuções subsequentes do crawler): escolha Crawl based on events (Rastreamento baseado em eventos) para usar notificações de eventos do Amazon S3 para seu crawler.
   - Include SQS ARN (Incluir ARN de SQS): especifique os parâmetros do armazenamento de dados, incluindo um ARN válido do SQS. (Por exemplo, `arn:aws:sqs:region:account:sqs`).
   - Include dead-letter SQS ARN (Incluir ARN de mensagens não entregues do SQS) (opcional): especifique um ARN válido de mensagens não entregues do SQS na Amazon. (Por exemplo, `arn:aws:sqs:region:account:deadLetterQueue`).
   - Selecione a opção Confirmar.



Usar criptografia com o crawler de eventos do Amazon S3

Esta seção descreve o uso de criptografia somente no SQS ou no SQS e no Amazon S3.

Tópicos

- Habilitar criptografia somente no SQS
- Habilitar a criptografia no SQS e no Amazon S3
- Perguntas frequentes

Habilitar criptografia somente no SQS

O Amazon SQS fornece criptografia em trânsito por padrão. Para adicionar a criptografia no lado do servidor (SSE) à sua fila, você pode anexar uma chave-mestra do cliente (CMK) no painel de edição. Isso significa que o SQS criptografará todos os dados do cliente em repouso em servidores SQS.

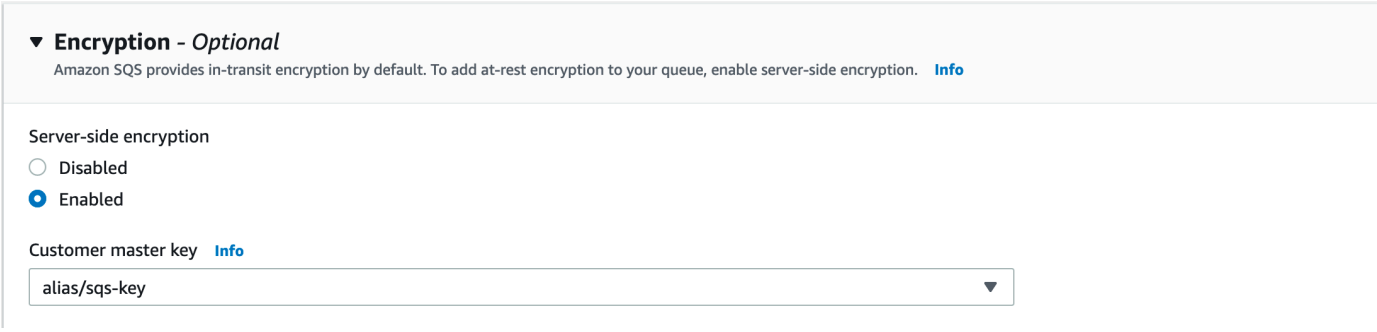
Criar uma chave-mestra de cliente (CMK)

1. Escolha Key Management Service (KMS) (Serviço de gerenciamento de chaves, KMS) > Customer Managed Keys (Chaves gerenciadas pelo cliente) > Create key (Criar chave).
2. Siga as etapas para adicionar seu próprio alias e descrição.
3. Adicione os respectivos perfis do IAM que você deseja que possam usar essa chave.
4. Na política de chaves, adicione outra declaração à lista "Statement" (Instrução) para que sua política de chaves personalizadas dê ao Amazon SNS permissões de uso de chaves suficientes.

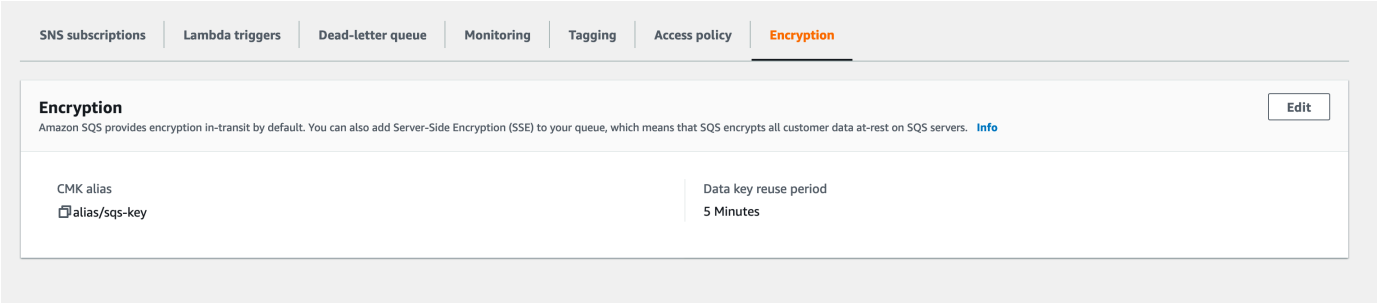
```
"Statement": [
       {
           "Effect": "Allow",
           "Principal": {
               "Service": "sns.amazonaws.com"
           },
           "Action": [
               "kms:GenerateDataKey",
               "kms:Decrypt"
           ],
           "Resource": "*"
       }
]
```

Habilite a criptografia no lado do servidor (SSE) na sua fila

1. Escolha a guia Amazon SQS > Queues (Filas) > sqs_queue_name > Encryption (Criptografia).
2. Escolha Edit (Editar) e role para baixo até o menu suspenso Encryption (Criptografia).
3. Selecione Enabled (Habilitado) para adicionar a SSE.
4. Selecione a CMK que você criou anteriormente, e não a chave padrão com o nome `alias/aws/sqs`.



Depois de adicionar isso, sua guia Encryption (Criptografia) será atualizada com a chave que você adicionou.



> **Note**
>
> O Amazon SQS exclui automaticamente as mensagens que estiverem em uma fila por mais tempo que o período de retenção máximo de mensagens. O período de retenção de mensagens padrão é de quatro dias. Para evitar eventos ausentes, altere o MessageRetentionPeriod (Período de retenção de mensagens) do SQS para o máximo de 14 dias.

Habilitar a criptografia no SQS e no Amazon S3

Habilitar a criptografia no lado do servidor (SSE) no SQS

1. Siga as etapas em the section called “Habilitar criptografia somente no SQS”.
2. Na última etapa da configuração da CMK, conceda ao Amazon S3 permissões de uso de chaves suficientes.

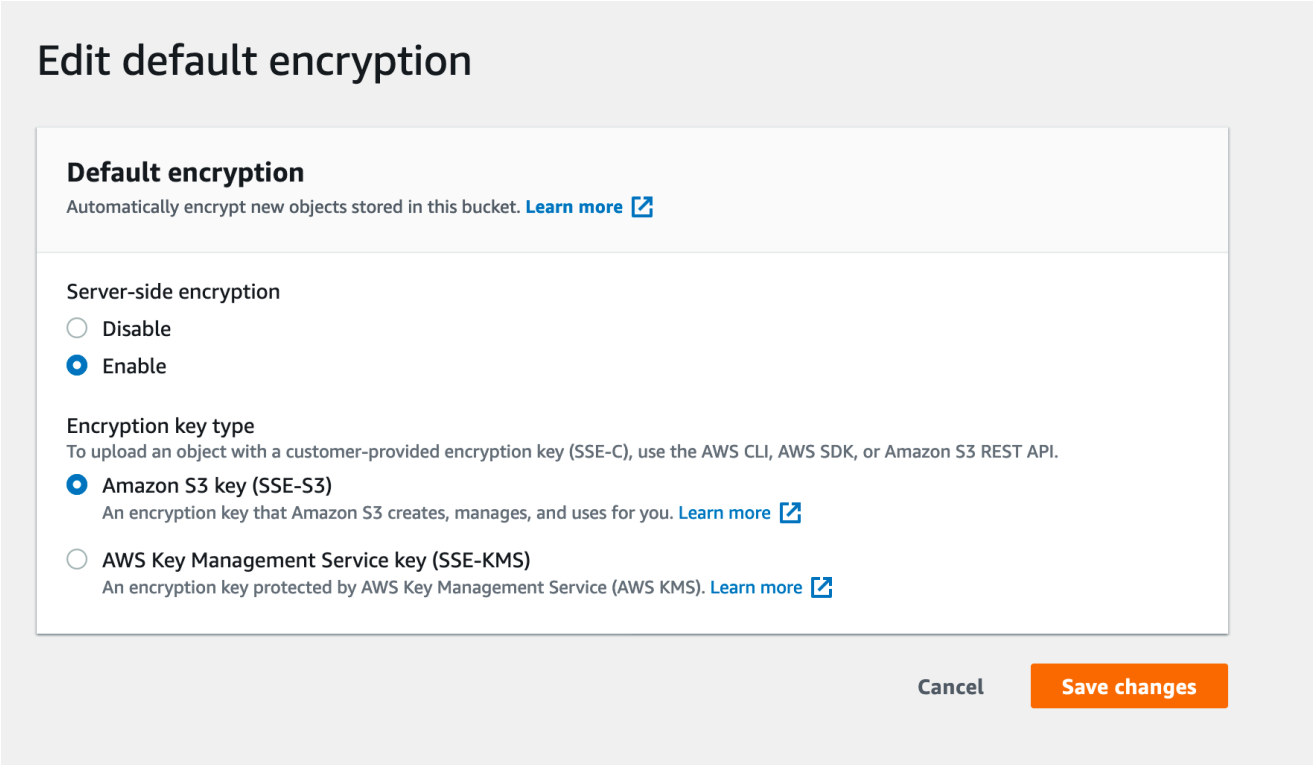
   Cole o seguinte na lista "Statement" (Declaração):

```
"Statement": [
       {
           "Effect": "Allow",
           "Principal": {
               "Service": "s3.amazonaws.com"
           },
           "Action": [
               "kms:GenerateDataKey",
               "kms:Decrypt"
           ],
           "Resource": "*"
       }
]
```

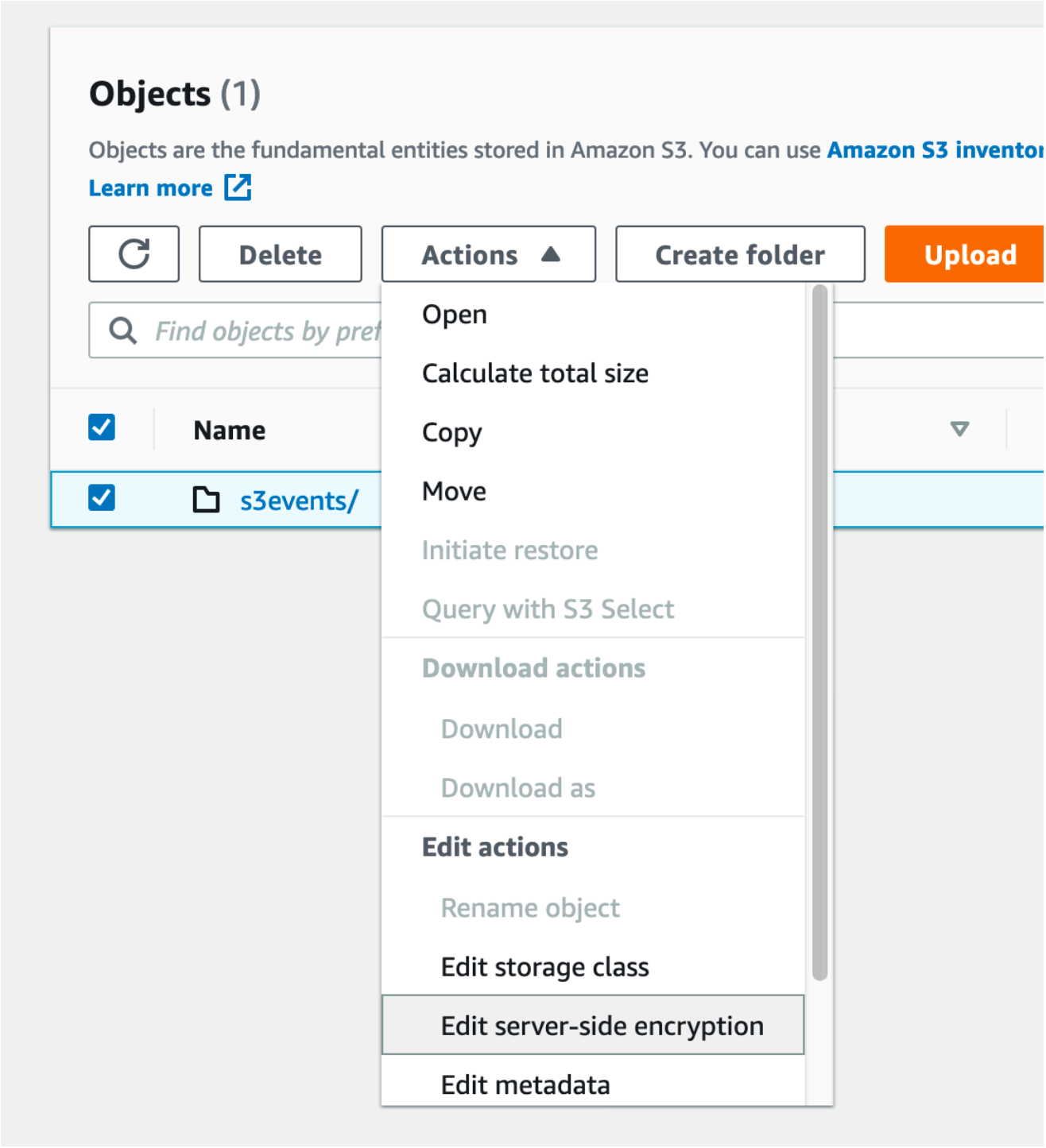
Habilitar a criptografia do lado do servidor (SSE) no bucket do Amazon S3

1. Siga as etapas em the section called “Habilitar criptografia somente no SQS”.
2. Execute um destes procedimentos:
   - Para habilitar o SSE para todo o seu bucket do S3, navegue até a guia Properties (Propriedades) no bucket de destino.

     Aqui você pode habilitar o SSE e escolher o tipo de criptografia que você gostaria de usar. O Amazon S3 fornece uma chave de criptografia que o Amazon S3 cria, gerencia e usa para você, ou você também pode escolher uma chave do KMS.



- Para habilitar o SSE em uma pasta específica, clique na caixa de seleção ao lado da pasta de destino e escolha Edit server-side encryption (Editar criptografia no lado do servidor) no menu suspenso Actions (Ações).



Perguntas frequentes

Por que as mensagens que eu publico no meu tópico do Amazon SNS não são entregues na minha fila inscrita do Amazon SQS com criptografia no lado do servidor (SSE) habilitada?

Verifique cuidadosamente se a fila do Amazon SQS está usando:

1. Uma chave-mestra do cliente (CMK) que seja gerenciada pelo cliente. Não a padrão fornecida pelo SQS.
2. Sua CMK de (1) inclui uma política de chaves personalizadas que dá ao Amazon SNS permissões de uso de chaves suficientes.

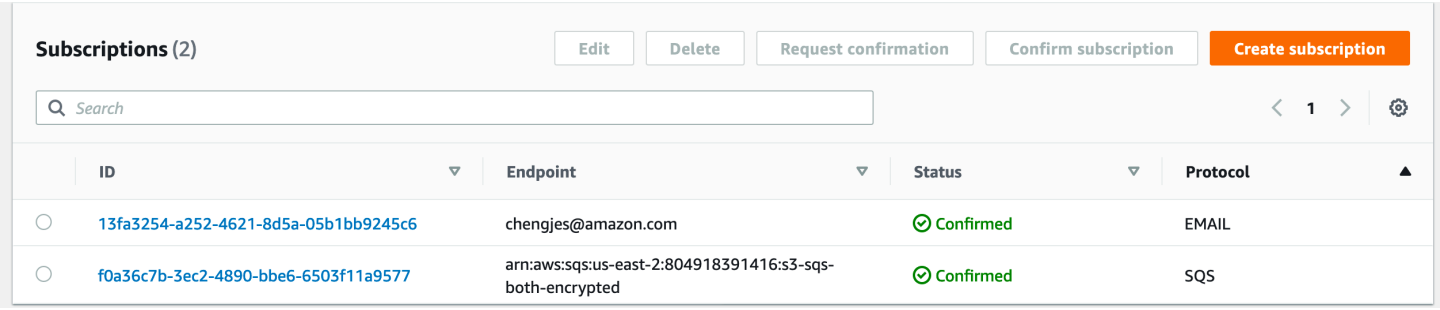
Para obter mais informações, consulte este artigo na Central de Conhecimento.

Eu me inscrevi em notificações por email, mas não recebo nenhuma atualização por email quando edito meu bucket do Amazon S3.

Verifique se você confirmou seu endereço de email clicando no link "Confirm subscription" (Confirmar assinatura) em seu email. Você pode conferir o status de sua confirmação verificando a tabela Subscriptions (Assinaturas) no seu tópico do SNS.

Escolha Amazon SNS > Topics (Tópicos) > `sns_topic_name` > Subscriptions table (Tabela de assinaturas).

Se você seguiu nosso script de pré-requisitos, você descobrirá que o `sns_topic_name` é igual ao seu `sqs_queue_name`. A aparência deve ser semelhante à seguinte:



Apenas algumas das pastas que adicionei estão aparecendo na minha tabela depois de habilitar a criptografia no lado do servidor na minha fila do SQS. Por que estou perdendo alguns parquets?

Se as alterações no bucket do Amazon S3 foram feitas antes de habilitar o SSE na fila do SQS, elas podem não ser coletadas pelo crawler. Para garantir que o crawl tenha passado por todas as atualizações para o bucket do S3, execute o crawler novamente no modo de listagem ("Crawl All Folders", Rastrear todas as pastas). Outra opção é começar de novo criando um novo crawler com eventos do S3 ativados.

Como impedir que o crawler altere um esquema existente

Se não quiser que um crawler substitua as atualizações feitas em campos existentes em uma definição de tabela do Amazon S3, escolha a opção no console Add new columns only (Adicionar

somente novas colunas) ou defina a opção de configuração `MergeNewColumns`. Isso se aplica a tabelas e partições, a menos que `Partitions.AddOrUpdateBehavior` seja substituído por `InheritFromTable`.

Se você não quiser que um esquema de tabela seja alterado quando um crawler for executado, defina a política de alteração de esquema como `LOG`. Você também pode definir uma opção de configuração que defina esquemas de partição para herdar da tabela.

Se você estiver configurando o crawler no console, poderá escolher as seguintes ações:

- Ignorar a alteração e não atualizar a tabela no Data Catalog
- Update all new and existing partitions with metadata from the table (Atualizar todas as partições novas e existentes com metadados da tabela)

Quando você configura o crawler usando a API, defina os seguintes parâmetros:

- Defina o campo `UpdateBehavior` na estrutura `SchemaChangePolicy` para `LOG`.
- Defina o campo `Configuration` com uma representação de string do objeto JSON a seguir na API do crawler, por exemplo:

```
{
   "Version": 1.0,
   "CrawlerOutput": {
      "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" }
   }
}
```

Como criar um esquema único para cada caminho de inclusão do Amazon S3

Por padrão, quando um crawler define tabelas para dados armazenados no Amazon S3, ele considera a compatibilidade dos dados e a similaridade do esquema. Os fatores de compatibilidade de dados que ele considera incluem: se os dados são do mesmo formato (por exemplo, JSON), do mesmo tipo de compactação (por exemplo, GZIP), a estrutura do caminho do Amazon S3 e outros atributos de dados. Similaridade de esquema é uma medida de até que ponto os esquemas de objetos do Amazon S3 separados são semelhantes.

Você pode configurar um crawler para `CombineCompatibleSchemas` em uma definição de tabela comum quando possível. Com essa opção, o crawler ainda considera a compatibilidade dos dados,

mas ignora a semelhança dos esquemas específicos ao avaliar objetos do Amazon S3 no caminho de inclusão especificado.

Se você estiver configurando o crawler no console, para combinar esquemas, selecione a opção de crawler Create a single schema for each S3 path (Criar um único esquema para cada caminho do S3).

Quando você configurar o crawler usando a API, defina a seguinte opção de configuração:

- Defina o campo `Configuration` com uma representação de string do objeto JSON a seguir na API do crawler, por exemplo:

```
{
   "Version": 1.0,
   "Grouping": {
      "TableGroupingPolicy": "CombineCompatibleSchemas" }
}
```

Para ajudar a ilustrar essa opção, suponha que você defina um crawler com um caminho de inclusão `s3://bucket/table1/`. Quando o crawler é executado, ele localiza dois arquivos JSON com as seguintes características:

- Arquivo 1: `S3://bucket/table1/year=2017/data1.json`
- Conteúdo do arquivo: `{“A”: 1, “B”: 2}`
- Esquema: `A:int, B:int`

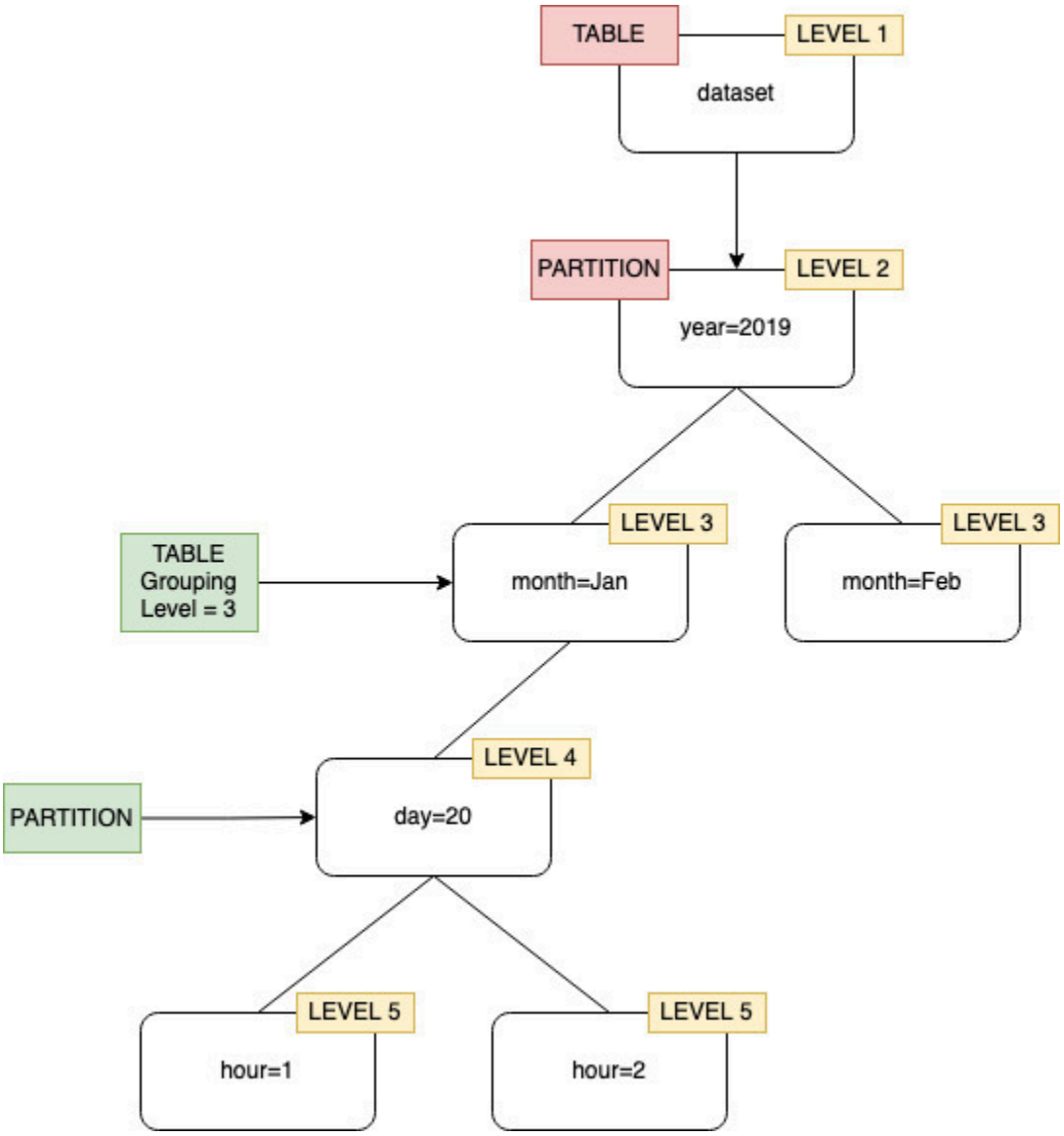
- Arquivo 2: `S3://bucket/table1/year=2018/data2.json`
- Conteúdo do arquivo: `{“C”: 3, “D”: 4}`
- Esquema: `C: int, D: int`

Por padrão, o crawler cria duas tabelas, chamadas `year_2017` e `year_2018`, porque os esquemas não são suficientemente semelhantes. No entanto, se a opção Create a single schema for each S3 path (Criar um único esquema para cada caminho do S3) for selecionada, e se os dados forem compatíveis, o crawler criará uma tabela. A tabela tem o esquema `A:int,B:int,C:int,D:int` e `partitionKey year:string`.

Como especificar o local da tabela e o nível de particionamento

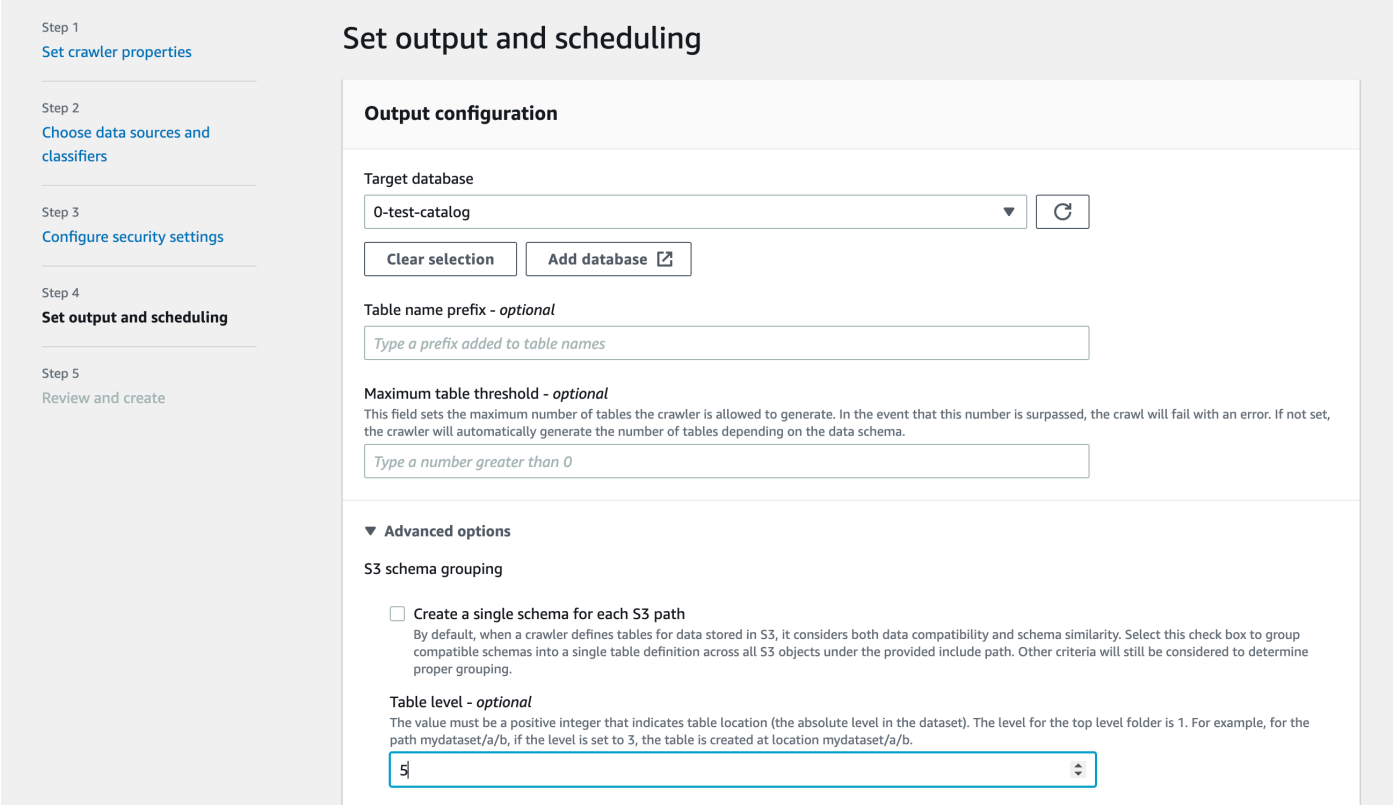
Por padrão, quando um crawler define tabelas para dados armazenados no Amazon S3, ele tenta mesclar esquemas e criar tabelas de nível superior (`year=2019`). Em alguns casos, você pode esperar que o crawler crie uma tabela para a pasta `month=Jan`, mas em vez disso, o crawler cria uma partição desde o momento em que uma pasta irmã (`month=Mar`) tenha sido mesclada na mesma tabela.

A opção de crawler no nível da tabela fornece a flexibilidade de informar ao crawler onde as tabelas estão localizadas e como você deseja que as partições sejam criadas. Quando você especifica um Table level (Nível da tabela), ela é criada nesse nível absoluto a partir do bucket do Amazon S3.



Ao configurar o crawler no console, você pode especificar um valor para a opção Table level (Nível da tabela) do Crawler. O valor deve ser um inteiro positivo que indica o local da tabela (o nível absoluto no conjunto de dados). O nível para a pasta de nível superior é um. Por exemplo, para o caminho `mydataset/year/month/day/hour`, se o nível for definido como três, a tabela será criada no local `mydataset/year/month`.

Console



API

Quando você configurar o crawler usando a API, defina o campo `Configuration` com uma representação de string do objeto JSON a seguir, por exemplo:

```
configuration = jsonencode(

{
   "Version": 1.0,
   "Grouping": {
            TableLevelConfiguration = 2
        }
})
```

CloudFormation

Neste exemplo, você define a opção Table level (Nível da tabela) disponível no console dentro do modelo do CloudFormation:

```
"Configuration": "{
    \"Version\":1.0,
    \"Grouping\":{\"TableLevelConfiguration\":2}
}"
```

Como especificar o número máximo de tabelas que o crawler pode criar

Opcionalmente, você pode especificar o número máximo de tabelas que o crawler pode criar especificando um `TableThreshold` por meio da CLI ou do console do AWS Glue. Se as tabelas detectadas pelo crawler durante o rastreamento forem maiores que esse valor de entrada, o rastreamento falhará e nenhum dado será gravado no Data Catalog.

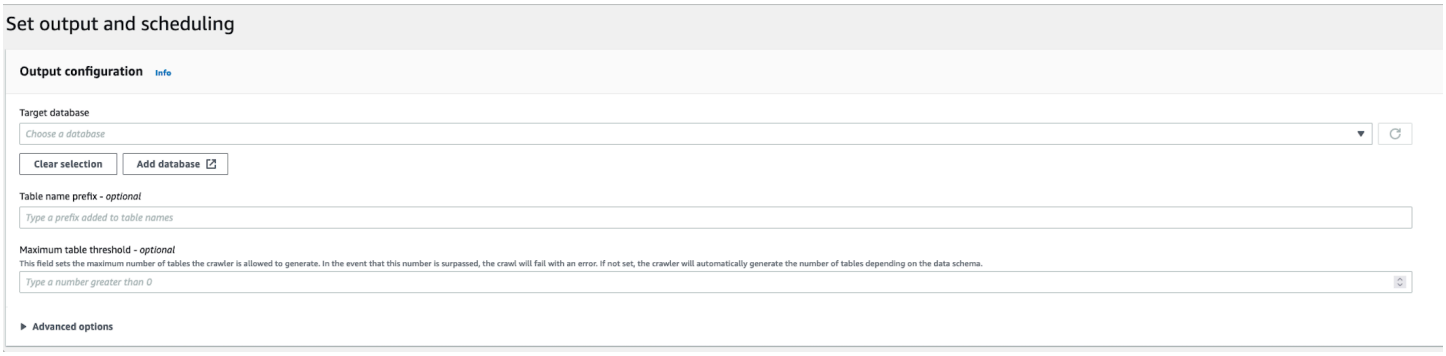
Esse parâmetro é útil quando as tabelas que seriam detectadas e criadas pelo crawler são muito maiores do que o esperado. Pode haver vários motivos para isso, por exemplo:

- Ao usar um trabalho do AWS Glue para preencher seus locais do Amazon S3, você pode acabar com arquivos vazios no mesmo nível de uma pasta. Nesses casos, quando você executa um crawler nesse local do Amazon S3, o crawler cria várias tabelas devido a arquivos e pastas presentes no mesmo nível.
- Se você não configurar `"TableGroupingPolicy": "CombineCompatibleSchemas"`, pode acabar com mais tabelas do que o esperado.

Você especifica o `TableThreshold` como um valor inteiro maior que 0. Esse valor é configurado para cada crawler. Ou seja, esse valor é considerado para cada rastreamento. Por exemplo: um crawler tem o valor `TableThreshold` definido como 5. Em cada rastreamento, o AWS Glue vai comparar o número de tabelas detectadas com esse valor limite da tabela (5). Se o número de tabelas detectadas for menor que 5, o AWS Glue vai gravar as tabelas no Data Catalog. Caso contrário, o rastreamento falhará sem gravar no Data Catalog.

Console

Para definir `TableThreshold` usando o console da AWS:



CLI

Para definir `TableThreshold` usando a AWS CLI:

```
"{"Version":1.0,
"CrawlerOutput":
{"Tables":{"AddOrUpdateBehavior":"MergeNewColumns",
"TableThreshold":5}}}";
```

As mensagens de erro são registradas para ajudar você a identificar os caminhos da tabela e a limpar seus dados. Exemplo: faça login em sua conta se o crawler falhar porque a contagem da tabela foi maior do que o valor limite da tabela fornecido:

```
Table Threshold value = 28, Tables detected - 29
```

No CloudWatch, registramos em log todas as localizações de tabelas detectadas como uma mensagem INFO. Um erro é registrado como o motivo da falha.

```
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.
```

Como especificar opções de configuração para um datastore do Delta Lake

Ao configurar um crawler para um datastore Delta Lake, é necessário especificar estes parâmetros de configuração:

Conexão

Opcionalmente, selecione ou adicione uma conexão de rede para usar com esse destino do Amazon S3. Para obter informações sobre conexões, consulte Conectar a dados.

Criar tabelas para consulta

Selecione como você deseja criar as tabelas do Delta Lake:

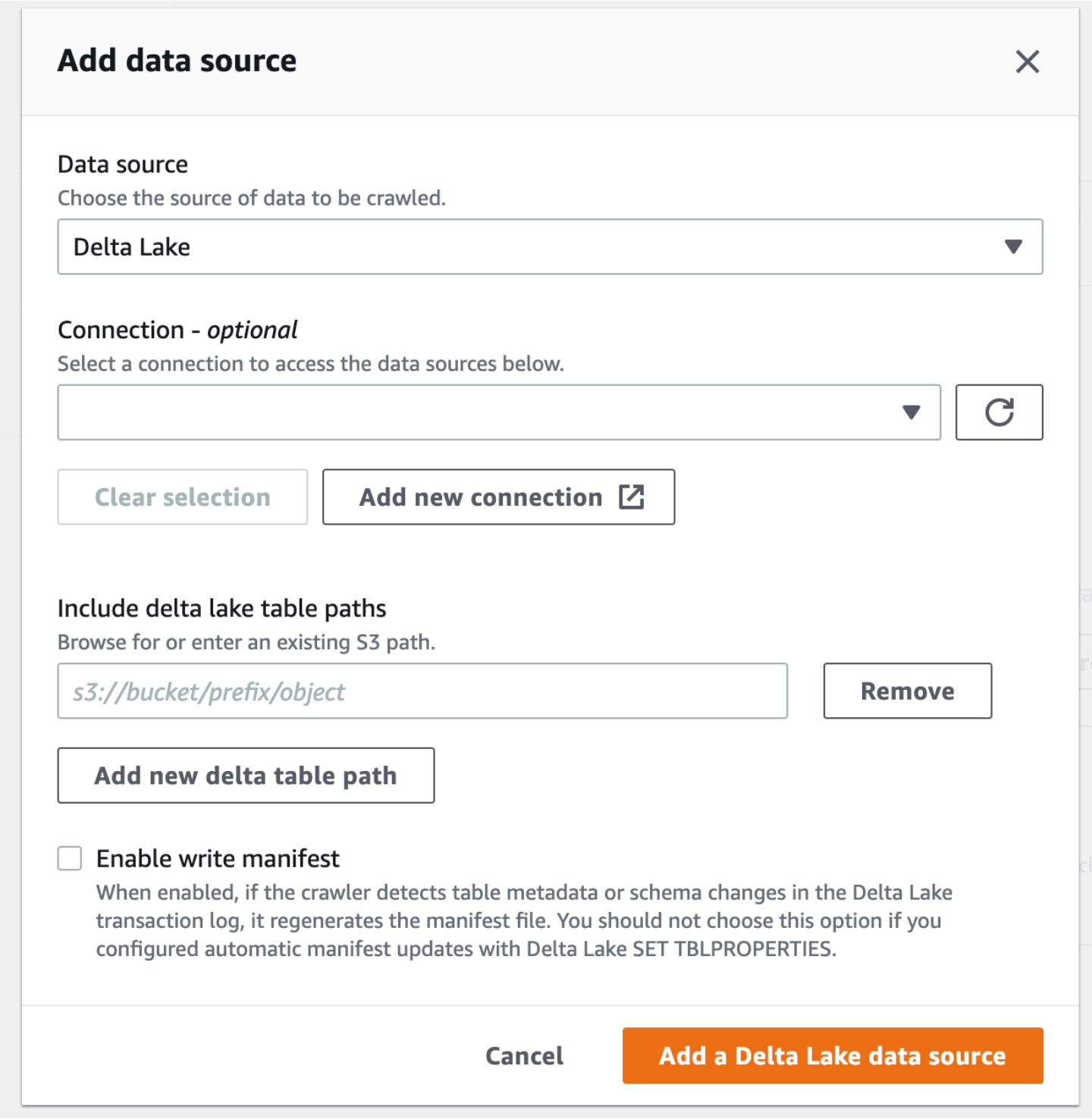
- Create Native tables (Criar tabelas nativas): permite a integração com mecanismos de consulta compatíveis com consulta direta ao log de transações do Delta.
- Create Symlink tables (Criar tabelas de link simbólico): Crie uma pasta manifesto de link simbólico com arquivos de manifesto particionados pelas chaves de partição, com base nos parâmetros de configuração especificados.

Habilite o manifesto de gravação (configurável somente se você tiver selecionado Criar tabelas de links simbólicos para uma fonte Delta Lake

Selecione se deseja detectar metadados da tabela ou alterações de esquema no log de transações do Delta Lake; fazer isso gera novamente o arquivo manifesto. Você não deverá escolher essa opção se tiver configurado uma atualização automática de manifesto com `SET TBLPROPERTIES` do Delta Lake.

Incluir caminhos das tabelas do Delta Lake

Especifique um ou mais caminhos do Amazon S3 para tabelas do Delta, como s3://*bucket*/*prefixo*/*objeto*.



Como configurar um crawler para usar credenciais do Lake Formation

Você pode configurar um crawler para usar credenciais do AWS Lake Formation para acessar um datastore do Amazon S3 ou uma tabela do Data Catalog com uma localização subjacente do Amazon S3 dentro da mesma Conta da AWS ou de outra Conta da AWS. Você pode configurar

uma tabela existente do Data Catalog como o destino do crawler, se o crawler e a tabela do Data Catalog residirem na mesma conta. No momento, é permitido ter somente um destino de catálogo com apenas uma tabela de catálogo ao usar uma tabela do Data Catalog como destino do crawler.

> **Note**
>
> Ao definir uma tabela do Data Catalog como um destino de crawler, certifique-se de que a localização subjacente da tabela do Data Catalog seja um local do Amazon S3. Os crawlers que usam credenciais do Lake Formation são compatíveis somente com destinos do Data Catalog com localizações subjacentes do Amazon S3.

Configuração exigida quando o crawler e a localização registrada do Amazon S3 ou a tabela do Data Catalog residem na mesma conta (crawling na conta)

Para permitir que o crawler acesse um datastore ou uma tabela do Data Catalog usando as credenciais do Lake Formation, você precisa registrar a localização dos dados com o Lake Formation. Além disso, o perfil do IAM do crawler deve ter permissões para ler os dados do destino no qual o bucket do Amazon S3 está registrado.

Você pode concluir as seguintes etapas de configuração usando o AWS Management Console ou a AWS Command Line Interface (AWS CLI).

AWS Management Console

1. Antes de configurar um crawler para acessar a fonte do crawler, registre a localização dos dados do datastore ou do Data Catalog com o Lake Formation. No console do Lake Formation (https://console.aws.amazon.com/lakeformation/), registre uma localização do Amazon S3 como a localização raiz do seu data lake na Conta da AWS na qual o crawler está definido. Para obter mais informações, consulte Registering an Amazon S3 location (Registrar um local do Amazon S3).
2. Conceda permissões de Data location (Localização de dados) ao perfil do IAM que é usado para a execução do crawler, de modo que o rastreador possa ler os dados do destino no Lake Formation. Para obter mais informações, consulte Granting data location permissions (same account) (Conceder permissões de localização de dados [mesma conta]).
3. Conceda permissões de acesso ao perfil do crawler (`Create`) para o banco de dados, que é especificado como o banco de dados de saída. Para obter mais informações, consulte Granting database permissions using the Lake Formation console and the named resource

method (Conceder permissões de banco de dados usando o console do Lake Formation e o método de recurso nomeado).

4. No console do IAM (https://console.aws.amazon.com/iam/), crie um perfil do IAM para o crawler. Adicione a política `lakeformation:GetDataAccess` ao perfil.
5. No console do AWS Glue (https://console.aws.amazon.com/glue/), ao configurar o crawler, selecione a opção Use Lake Formation credentials for crawling Amazon S3 data source (Usar credenciais do Lake Formation para crawling da fonte de dados do Amazon S3).

> **Note**
>
> O campo accountId é opcional para crawling na conta.

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": ""
  }
}'
```

Configuração exigida quando o crawler e a localização registrada do Amazon S3 residem em contas diferentes (crawling entre contas)

Para permitir que o crawler acesse um datastore em uma conta diferente usando as credenciais do Lake Formation, primeiro você deve registrar a localização de dados do Amazon S3 com o Lake Formation. Em seguida, você concede permissões de localização de dados à conta do crawler conforme as etapas a seguir.

Você pode concluir as seguintes etapas usando o AWS Management Console ou a AWS CLI.

AWS Management Console

1. Na conta em que a localização do Amazon S3 está registrada (conta B):
   a. Registre um caminho do Amazon S3 no Lake Formation. Para obter mais informações, consulte Registering an Amazon S3 location (Registrando um local do Amazon S3).
   b. Conceda permissões de Data location (Localização de dados) para a conta (conta A) na qual o crawler será executado. Para obter mais informações, consulte Granting data location permissions (Conceder permissões de localização de dados).
   c. Crie um banco de dados vazio no Lake Formation com a localização subjacente como o local de destino do Amazon S3. Para obter mais informações, consulte Creating a database (Criar um banco de dados).
   d. Conceda à conta A (a conta na qual o crawler será executado) acesso ao banco de dados que você criou na etapa anterior. Para obter mais informações, consulte Granting database permissions (Conceder permissões de banco de dados).

2. Na conta em que o crawler é criado e será executado (conta A):

   a. Ao usar o console da AWS RAM, aceite o banco de dados que foi compartilhado da conta externa (conta B). Para obter mais informações, consulte Aceitar um convite de compartilhamento de recursos do AWS Resource Access Manager.

   b. Crie um perfil do IAM para o crawler. Adicione a política `lakeformation:GetDataAccess` ao perfil.

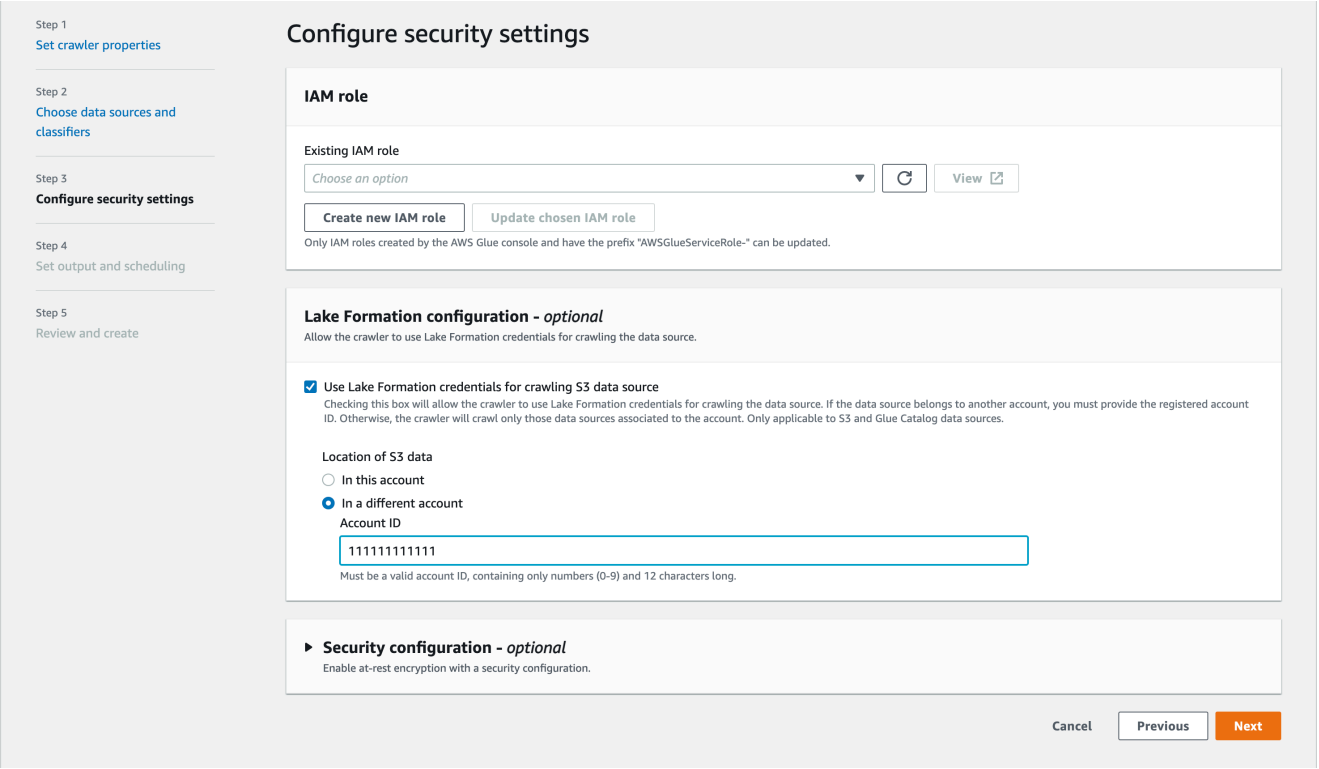
   c. No console do Lake Formation (https://console.aws.amazon.com/lakeformation/), conceda permissões de Data location (Localização dos dados) na localização de destino do Amazon S3 para o perfil do IAM usado para a execução do crawler, de modo que o crawler possa ler os dados do destino no Lake Formation. Para obter mais informações, consulte Granting data location permissions (Conceder permissões de localização de dados).

   d. Crie um link de recurso no banco de dados compartilhado. Para obter mais informações, consulte Create a resource link (Criar um link de recurso).

   e. Conceda permissões de acesso ao perfil do crawler (`Create`) no banco de dados compartilhado e (`Describe`) no link do recurso. O link do recurso é especificado na saída do crawler.

   f. No console do AWS Glue (https://console.aws.amazon.com/glue/), ao configurar o crawler, selecione a opção Use Lake Formation credentials for crawling Amazon S3 data source (Usar credenciais do Lake Formation para crawling da fonte de dados do Amazon S3).

      Para crawling entre contas, especifique o ID da Conta da AWS na qual a localização de destino do Amazon S3 está registrada no Lake Formation. O campo accountId é opcional para crawling na conta.



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

- Um crawler que use credenciais do Lake Formation é compatível somente com destinos do Amazon S3 e do Data Catalog.
- Para destinos que utilizam o fornecimento de credenciais do Lake Formation, as localizações subjacentes do Amazon S3 devem pertencer ao mesmo bucket. Por exemplo, os clientes podem usar vários destinos (s3://bucket1/folder1, s3://bucket1/folder2), desde que todos os locais de destino estejam no mesmo bucket (bucket1). Não é permitido especificar buckets diferentes (s3://bucket1/folder1, s3://bucket2/folder2).
- No momento, só é permitido ter um destino de catálogo com uma só tabela de catálogo para crawlers de destino do Data Catalog.

Tutorial: Adicionar um crawler do AWS Glue

Para este cenário do AWS Glue, será solicitado que você analise os dados de chegada das principais transportadoras aéreas para calcular a popularidade dos aeroportos de partida mês a mês.

Você tem dados de voos do ano de 2016 no formato CSV armazenados no Amazon S3. Antes de transformar e analisar seus dados, você cataloga seus metadados no AWS Glue Data Catalog.

Neste tutorial, vamos adicionar um crawler que infere metadados desses logs de voos no Amazon S3 e cria uma tabela no Data Catalog.

Tópicos

Pré-requisitos

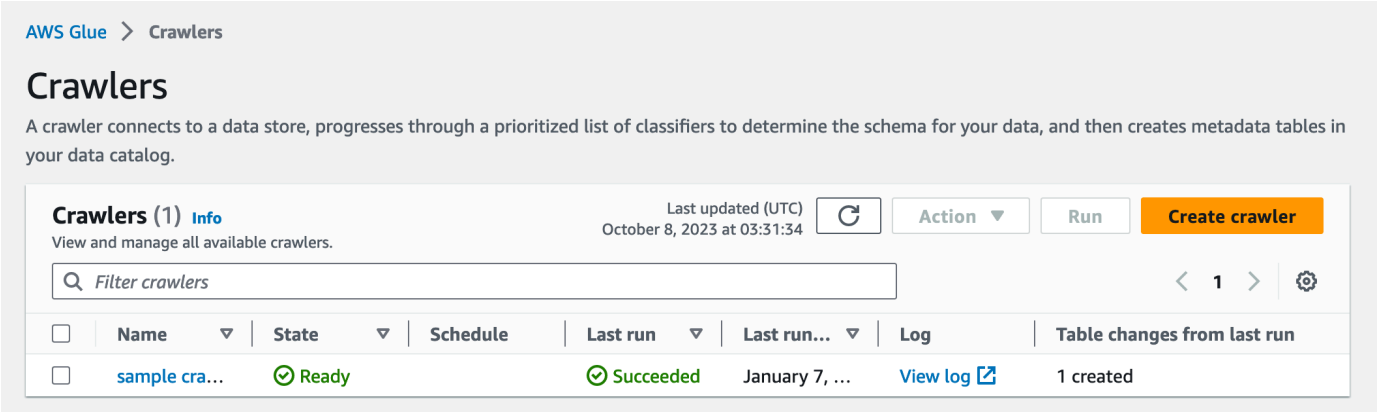
Este tutorial pressupõe que você tem uma conta da AWS e acesso ao AWS Glue.

Etapa 1: Adicionar um crawler

Use estas etapas para configurar e executar um crawler que extrai os metadados de um arquivo CSV armazenado no Amazon S3.

Para criar um crawler que lê arquivos armazenados no Amazon S3

1. No console de serviço do AWS Glue, no menu do lado esquerdo, selecione Crawlers.
2. Na página Crawlers, escolha Create crawler (Criar crawler). Isso inicia uma série de páginas que solicitam os detalhes do crawler.



3. No campo de nome do crawler, insira **`Flights Data Crawler`** e, em seguida, escolha Next (Próximo).

Crawlers invocam classificadores para inferir o esquema dos seus dados. Este tutorial usa o classificador integrado para CSV por padrão.

4. Para o tipo de origem do crawler, escolha Data stores (Armazenamentos de dados) e, em seguida, Next (Próximo).
5. Agora, vamos apontar o crawler para os seus dados. Na página Add a data store (Adicionar um armazenamento de dados), escolha o armazenamento de dados do Amazon S3. Este tutorial não usa uma conexão, então deixe o campo Connection (Conexão) em branco, se estiver visível.

   Na opção Crawl data in (Rastrear dados em), escolha Specified path in another account (Caminho especificado em outra conta). Em Include path (Incluir o caminho), insira o caminho em que o crawler pode encontrar os dados de voos, que é **`s3://crawler-public-us-east-1/flight/2016/csv`**. Depois de inserir o caminho, o título desse campo muda para Include path (Incluir o caminho). Escolha Próximo.
6. É possível rastrear vários armazenamentos de dados com um único crawler. No entanto, neste tutorial, estamos usando apenas um único armazenamento de dados, então escolha No (Não) e, depois, escolha Next (Próximo).
7. O crawler precisa de permissões para acessar o armazenamento de dados e criar objetos no AWS Glue Data Catalog. Para configurar essas permissões, escolha Create an IAM role (Criar uma função do IAM). O nome da função do IAM começa com `AWSGlueServiceRole-` e, no campo, insira a última parte do nome da função. Insira **`CrawlerTutorial`** e escolha Next (Próximo).

   > **Note**
   >
   > Para criar uma função do IAM, seu usuário da AWS deve ter as permissões `CreateRole`, `CreatePolicy` e `AttachRolePolicy`.

   O assistente cria uma função do IAM denominada `AWSGlueServiceRole-CrawlerTutorial`, anexa a política gerenciada pela AWS `AWSGlueServiceRole` a essa função e adiciona uma política embutida que permite acesso de leitura ao local do Amazon S3 `s3://crawler-public-us-east-1/flight/2016/csv`.
8. Crie uma programação para o crawler. Em Frequency (Frequência), escolha Run on demand (Executar sob demanda) e, depois, escolha Next (Próximo).

9. Os crawlers criam tabelas no Data Catalog. As tabelas são contidas em um banco de dados no Data Catalog. Primeiro, escolha Add database (Adicionar banco de dados) para criar um banco de dados. Na janela pop-up, insira **`test-flights-db`** no nome do banco de dados e escolha Create (Criar).

   Em seguida, insira **`flights`** em Prefix added to tables (Prefixo adicionado às tabelas). Use os valores padrão para o restante das opções e escolha Next (Próximo).
10. Verifique as opções que você escolheu no assistente Add crawler (Adicionar crawler). Se você vir algum erro, pode escolher Back (Voltar) para retornar às páginas anteriores e fazer alterações.

    Depois de revisar as informações, escolha Finish (Finalizar) para criar o crawler.

Etapa 2: Executar o crawler

Depois de criar um crawler, o assistente o envia para a página de exibição de crawlers. Como você cria o crawler com uma programação sob demanda, você tem a opção de executá-lo.

Para executar o crawler

1. O banner próximo à parte superior dessa página permite que você saiba que o crawler foi criado e pergunta se você deseja executá-lo agora. Escolha Run it now? (Executá-lo agora?) para executar o crawler.

   O banner muda para mostrar as mensagens “Attempting to run” (Tentativa de execução) e “Running” (Em execução) para o crawler. Depois que o crawler começa a ser executado, o banner desaparece e a exibição do crawler é atualizada para mostrar o status “Starting” (Iniciando) para o crawler. Após um minuto, você pode clicar no ícone Refresh (Atualizar) para atualizar o status do crawler exibido na tabela.
2. Quando o crawler for concluído, será exibido um novo banner que descreve as alterações feitas por ele. Você pode escolher o link test-flights-db para exibir os objetos do Data Catalog.

Etapa 3: Exibir objetos do AWS Glue Data Catalog

O crawler lê dados no local de origem e cria tabelas no Data Catalog. A tabela é uma definição de metadados que representa seus dados, incluindo o esquema. As tabelas no Data Catalog não contêm dados. Em vez disso, você usa essas tabelas como uma origem ou destino em uma definição de trabalho.

Para exibir os objetos do Data Catalog criados pelo crawler

1. No painel de navegação do lado esquerdo, em Data Catalog, escolha Databases (Bancos de dados). Aqui você pode ver o banco de dados `flights-db` criado pelo crawler.
2. No painel de navegação do lado esquerdo, em Data Catalog e abaixo de Databases (Bancos de dados), escolha Tables (Tabelas). Aqui, você pode ver a tabela `flightscsv` criada pelo crawler. Se você escolher o nome da tabela, poderá exibir as configurações, os parâmetros e as propriedades dela. Ao rolar para baixo nessa exibição, você pode ver o esquema, que são informações sobre as colunas e os tipos de dados da tabela.
3. Se escolher View partitions (Exibir partições) na página de exibição da tabela, poderá ver as partições criadas para os dados. A primeira coluna é a chave de partição.

Definir metadados manualmente

O Catálogo de Dados do AWS Glue é um repositório central que armazena metadados sobre suas fontes de dados e conjuntos de dados. Embora um crawler possa obter e preencher automaticamente os metadados das fontes de dados compatíveis, há alguns cenários em que talvez seja necessário definir os metadados manualmente no Catálogo de Dados:

- Formatos de dados incompatíveis: se você tiver fontes de dados que não são compatíveis com o crawler, será necessário definir manualmente os metadados dessas fontes de dados no Catálogo de Dados.
- Requisitos de metadados personalizados: o Crawler do AWS Glue infere metadados com base em regras e convenções predefinidas. Se você tiver requisitos específicos de metadados que não são cobertos pelos metadados Crawler do AWS Glue inferidos, você poderá definir manualmente os metadados para atender às suas necessidades
- Governança e padronização de dados: em alguns casos, talvez você queira ter mais controle sobre as definições de metadados por motivos de governança, conformidade ou segurança de dados. A definição manual de metadados permite garantir que os metadados estejam de acordo com os padrões e políticas da sua organização.
- Espaço reservado para futura ingestão de dados: se houver fontes de dados que não estão imediatamente disponíveis ou acessíveis, você poderá criar tabelas de esquema vazias como espaços reservados. Quando as fontes de dados se tornarem disponíveis, você poderá preencher as tabelas com os dados reais, mantendo a estrutura predefinida.

Para definir metadados manualmente, você pode usar o console do AWS Glue, o console do Lake Formation, a API do AWS Glue ou o a AWS Command Line Interface (AWS CLI). É possível criar bancos de dados, tabelas e partições e especificar propriedades de metadados, como nomes de colunas, tipos de dados, descrições e outros atributos.

Criar bancos de dados

Os bancos de dados são usados para organizar tabelas de metadados no AWS Glue. Ao definir uma tabela no AWS Glue Data Catalog, você a adiciona a um banco de dados. Uma tabela pode estar somente em um banco de dados por vez.

Seu banco de dados pode conter tabelas que definem dados de diferentes armazenamentos de dados. Esses dados podem incluir objetos no Amazon Simple Storage Service (Amazon S3) e tabelas relacionais no Amazon Relational Database Service.

> **Note**
>
> Quando você exclui um banco de dados do AWS Glue Data Catalog, todas as tabelas contidas nele também são excluídas.

Para visualizar a lista de bancos de dados, faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/. Escolha Databases e, em seguida, selecione o nome do banco de dados na lista para ver os detalhes dele.

Na guia Databases no console do AWS Glue, você pode adicionar, editar e excluir bancos de dados:

- Para criar um novo banco de dados, escolha Add database e forneça um nome e uma descrição. Para compatibilidade com outros armazenamentos de metadados, como o Apache Hive, as letras do nome são transformadas em minúsculas.

  > **Note**
  >
  > Se você planeja acessar o banco de dados a partir do Amazon Athena, forneça um nome somente com caracteres alfanuméricos e sublinhados. Para obter mais informações, consulte Nomes de tabelas, bancos de dados e colunas.

- Para editar a descrição de um banco de dados, marque a caixa de seleção ao lado do nome do banco de dados e escolha Edit (Editar).

- Para excluir um banco de dados, marque a caixa de seleção ao lado do nome do banco de dados e escolha Remove.
- Para exibir a lista de tabelas contidas no banco de dados, escolha o nome do banco de dados e as propriedades do banco de dados exibirão todas as tabelas no banco de dados.

Para alterar o banco de dados em que o crawler escreve, você precisa alterar a definição desse crawler. Para ter mais informações, consulte Usar crawlers para preencher o catálogo de dados .

Links de recursos de bancos de dados

> O console do AWS Glue passou por uma atualização recente. A versão atual do console não é compatível com links de recursos de banco de dados.

O Data Catalog também pode conter links do recurso para bancos de dados. Um link de recurso de banco de dados é um link para um banco de dados local ou compartilhado. No momento, você pode criar links de recursos somente no AWS Lake Formation. Depois de criar um link de recurso para um banco de dados, você pode usar o nome do link de recurso onde quer que você use o nome do banco de dados. Junto com bancos de dados que você possui ou que são compartilhados com você, links de recursos de banco de dados são retornados por `glue:GetDatabases()` e aparecem como entradas na página Databases (Bancos de dados) do console do AWS Glue.

O Data Catalog também pode conter links do recurso para tabelas.

Para obter mais informações sobre links de recursos, consulte Criar links de recursos no Guia do desenvolvedor do AWS Lake Formation.

Criar tabelas

Embora a execução de um crawler seja o método recomendado para fazer o inventário dos dados em seus armazenamentos de dados, você pode adicionar tabelas de metadados ao AWS Glue Data Catalog manualmente. Essa abordagem permite que você tenha mais controle sobre as definições de metadados e as personalize de acordo com seus requisitos específicos.

Você também pode adicionar definições de tabela manualmente ao Catálogo de Dados das seguintes formas:

- Use o console do AWS Glue para criar manualmente uma tabela no AWS Glue Data Catalog. Para ter mais informações, consulte Trabalhar com tabelas no console do AWS Glue.

- Use a operação `CreateTable` em AWS Glue API para criar uma tabela no AWS Glue Data Catalog. Para ter mais informações, consulte CreateTable ação (Python: create_table).
- Use modelos do AWS CloudFormation. Para ter mais informações, consulte AWS CloudFormation para AWS Glue.

Ao definir uma tabela manualmente usando o console ou uma API, você especifica o esquema da tabela e o valor de um campo de classificação que indica o tipo e o formato dos dados na fonte de dados. Se um crawler criar a tabela, o esquema e o formato dos dados serão determinados por um classificador integrado ou um classificador personalizado. Para obter mais informações sobre como criar uma tabela usando o console do AWS Glue, consulte Trabalhar com tabelas no console do AWS Glue.

Tópicos

- Partições de tabela
- Links de recursos de tabela
- Atualizar tabelas do Data Catalog criadas manualmente usando crawlers
- Propriedades da tabela do Data Catalog
- Trabalhar com tabelas no console do AWS Glue
- Trabalhar com índices de partição no AWS Glue

Partições de tabela

Uma definição de tabela do AWS Glue de uma pasta do Amazon Simple Storage Service (Amazon S3) pode descrever uma tabela particionada. Por exemplo, para melhorar a performance da consulta, uma tabela particionada pode separar dados mensais em diferentes arquivos usando o nome do mês como uma chave. No AWS Glue, as definições de tabela incluem a chave de particionamento de uma tabela. Quando o AWS Glue avalia os dados nas pastas do Amazon S3 para catalogar uma tabela, ele determina se foi adicionada uma tabela individual ou particionada.

Você pode criar índices de partição em uma tabela para buscar um subconjunto das partições em vez de carregar todas as partições na tabela. Para obter mais informações sobre como trabalhar com índices de partição, consulte Trabalhar com índices de partição no AWS Glue.

Todas as condições a seguir precisam ser verdadeiras para que o AWS Glue crie uma tabela particionada para uma pasta do Amazon S3:

- Os esquemas dos arquivos são semelhantes, conforme determinado pelo AWS Glue.
- O formato de dados dos arquivos é o mesmo.
- A compressão de dados dos arquivos é a mesma.

Por exemplo, você pode possuir um bucket do Amazon S3 chamado `my-app-bucket`, onde armazena dados de vendas de aplicações iOS e Android. Os dados são particionados por ano, mês e dia. Os arquivos de dados para vendas de aplicativos iOS e Android têm o mesmo esquema, formato de dados e formato de compactação. No AWS Glue Data Catalog, o crawler do AWS Glue cria uma definição de tabela com chaves de particionamento por ano, mês e dia.

A seguinte listagem do Amazon S3 de `my-app-bucket` mostra algumas das partições. O símbolo = é usado para atribuir valores de chave de partição.

```
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
```

Links de recursos de tabela

> O console do AWS Glue passou por uma atualização recente. A versão atual do console não é compatível com links de recursos de tabela.

O Data Catalog também pode conter links de recursos para tabelas. Um link de recurso de tabela é um link para uma tabela local ou compartilhada. No momento, você pode criar links de recursos somente no AWS Lake Formation. Depois de criar um link de recurso para uma tabela, você pode usar o nome do link de recurso onde quer que você use o nome da tabela. Junto com tabelas que você possui ou que são compartilhadas com você, links de recursos de tabelas são retornados por `glue:GetTables()` e aparecem como entradas na página Tables (Tabelas) do console do AWS Glue.

O Data Catalog também pode conter links de recursos de banco de dados.

Para obter mais informações sobre links de recursos, consulte Criar links de recursos no Guia do desenvolvedor do AWS Lake Formation.

Atualizar tabelas do Data Catalog criadas manualmente usando crawlers

Você pode querer criar tabelas do AWS Glue Data Catalog manualmente e mantê-las atualizadas com crawlers do AWS Glue. Os crawlers em execução em uma programação podem adicionar novas partições e atualizar as tabelas com qualquer alteração de esquema. Isso também se aplica a tabelas migradas de um metastore do Apache Hive.

Para fazer isso, ao definir um crawler, em vez de especificar um ou mais armazenamentos de dados como a fonte de um crawl, você especifica uma ou mais tabelas do Data Catalog existentes. Em seguida, o crawler rastreia os armazenamentos de dados especificados pelas tabelas de catálogo. Nesse caso, nenhuma tabela nova é criada; em vez disso, suas tabelas criadas manualmente são atualizadas.

Veja a seguir os outros motivos pelos quais você pode querer criar manualmente tabelas de catálogo e especificar tabelas de catálogo como a origem do crawler:

- Você deseja escolher o nome da tabela de catálogo e não depender do algoritmo de nomenclatura da tabela de catálogo.
- Você deseja impedir que novas tabelas sejam criadas caso arquivos com um formato que possa interromper a detecção de partição sejam salvos por engano no caminho da fonte de dados.

Para ter mais informações, consulte Etapa 2: escolher as fontes de dados e os classificadores.

Propriedades da tabela do Data Catalog

As propriedades ou parâmetros da tabela, como são conhecidos na AWS CLI, são strings de chave e valor não validadas. Você pode definir suas próprias propriedades na tabela para permitir o uso do Data Catalog fora do AWS Glue. Outros serviços que usam o Data Catalog também podem fazer isso. AWS Glue define algumas propriedades da tabela ao executar trabalhos ou rastreadores. Salvo indicação em contrário, essas propriedades são para uso interno. Não podemos garantir que elas continuarão existindo em sua forma atual nem que o produto se comportará conforme esperado se essas propriedades forem alteradas manualmente.

Para obter mais informações sobre as propriedades da tabela definidas pelos crawlers do AWS Glue, consulte the section called “Parâmetros definidos nas tabelas do Data Catalog pelo crawler”.

Trabalhar com tabelas no console do AWS Glue

Uma tabela no AWS Glue Data Catalog é a definição de metadados que representa os dados em um datastore. Você cria tabelas quando executa um crawler ou manualmente no console do AWS Glue. A lista Tables (Tabelas) no console do AWS Glue exibe valores dos metadados da sua tabela. Você usa definições de tabela para especificar fontes e destino ao criar trabalhos de ETL (extração, transformação e carregamento).

> **Note**
>
> Com as mudanças recentes no console de gerenciamento da AWS, pode ser necessário modificar os perfis do IAM existentes para que tenham a permissão de `SearchTables`. Para a criação de um novo perfil, a permissão da API `SearchTables` já foi adicionada como padrão.

Para começar, faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/. Escolha a guia Tables e use o botão Add tables para criar tabelas com um crawler ou digitando atributos manualmente.

Adicionar tabelas ao console

Para usar um crawler para adicionar tabelas, escolha Add tables, Add tables using a crawler. Em seguida, siga as instruções no assistente Add crawler. Quando o crawler for executado, as tabelas serão adicionadas ao AWS Glue Data Catalog. Para ter mais informações, consulte Usar crawlers para preencher o catálogo de dados .

Se você conhecer os atributos necessários para criar uma definição de tabela do Amazon Simple Storage Service (Amazon S3) no Data Catalog, você poderá criá-la com o assistente de tabela. Escolha Add tables, Add table manually a siga as instruções no assistente Add table.

Ao adicionar uma tabela manualmente usando o console, considere o seguinte:

- Se você planeja acessar a tabela a partir do Amazon Athena, forneça um nome somente com caracteres alfanuméricos e sublinhados. Para obter mais informações, consulte Nomes do Athena.
- O local dos seus dados de origem deve ser um caminho do Amazon S3.
- O formato dos dados deve corresponder a um dos formatos listados no assistente. A classificação correspondente, SerDe e outras propriedades da tabela serão preenchidas automaticamente com base no formato escolhido. Você pode definir tabelas com os seguintes formatos:

Avro

Formato binário JSON do Apache Avro.

CSV

Valores separados por vírgula (CSV). Você também especifica o delimitador como vírgula, pipe, ponto e vírgula, tabulação ou Ctrl+A.

JSON

JavaScript Object Notation.

XML

Formato Extensible Markup Language. Especifique a tag XML que define uma linha nos dados. As colunas são definidas nas tags de linha.

Parquet

Armazenamento em colunas no Apache Parquet.

ORC

Formato Optimized Row Columnar (ORC). Um formato criado para armazenar dados do Hive com eficiência.

- Você pode definir uma chave de partição para a tabela.
- Atualmente, as tabelas particionadas que você cria com o console não podem ser usadas em trabalhos de ETL.

Atributos da tabela

Veja a seguir alguns atributos importantes da sua tabela:

Nome

O nome é determinado quando a tabela é criada, e você não pode alterá-la. Você faz referência a um nome de tabela em muitas operações do AWS Glue.

Banco de dados

O objeto do contêiner onde a sua tabela reside. Este objeto contém uma organização das suas tabelas que existe dentro do AWS Glue Data Catalog e pode diferir de uma organização no seu

datastore. Quando você exclui um banco de dados, todas as tabelas contidas nele também são excluídas do Data Catalog.

Descrição

A descrição da tabela. Você pode escrever uma descrição para ajudá-lo a entender o conteúdo da tabela.

Formato da tabela

Especifique a criação de uma tabela padrão do AWS Glue ou de uma tabela no formato do Apache Iceberg.

Habilitar compactação

Escolha Habilitar compactação para compactar pequenos objetos do Amazon S3 na tabela em objetos maiores.

IAM role (Perfil do IAM)

Para executar a compactação, o serviço assume um perfil do IAM em seu nome. Você pode escolher um perfil do IAM usando o menu suspenso. Certifique-se de que o perfil tenha as permissões necessárias para habilitar a compactação.

Para saber mais sobre as permissões necessárias para o perfil do IAM, consulte Pré-requisitos de otimização de tabelas .

Local

O ponteiro para o local dos dados em um datastore que esta definição de tabela representa.

Classificação

Um valor de categorização fornecido quando a tabela foi criada. Normalmente, ele é escrito quando um crawler é executado e especifica o formato dos dados da fonte.

Última atualização

A hora e a data (UTC) em que esta tabela foi atualizada no Data Catalog.

Data adicionada

A hora e a data (UTC) em que esta tabela foi adicionada ao Data Catalog.

Preterido

Se o AWS Glue descobrir que uma tabela do Data Catalog não existe mais no seu datastore original, ele marcará essa tabela como defasada no catálogo de dados. Se você executar

um trabalho que faz referência a uma tabela obsoleta, ele falhará. Edite trabalhos que fazem referência a tabelas obsoletas para removê-las como fontes e destinos. Recomendamos que você elimine tabelas obsoletas quando elas não forem mais necessárias.

Conexão

Se o AWS Glue exigir conexão com seu datastore, o nome da conexão será associado à tabela.

Exibir e editar detalhes da tabela

Para ver os detalhes de uma tabela existente, escolha o nome dela na lista e, em seguida, Action, View details.

Esses detalhes incluem propriedades da sua tabela e do seu esquema. Essa exibição mostra o esquema da tabela, incluindo os nomes de colunas na ordem definida para a tabela, os tipos de dados e as colunas de chaves para partições. Se uma coluna for de um tipo complexo, você poderá escolher View properties para exibir detalhes da estrutura desse campo, como mostrado no exemplo a seguir:

```
{
"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 obter mais informações sobre as propriedades de uma tabela, como `StorageDescriptor`, consulte StorageDescriptor estrutura.

Para alterar o esquema de uma tabela, escolha Edit schema para adicionar e remover colunas, alterar nomes de colunas e alterar tipos de dados.

Para comparar diferentes versões de uma tabela, incluindo seu esquema, escolha Compare versions para ver uma comparação lado-a-lado de duas versões do esquema para uma tabela. Para ter mais informações, consulte Comparar versões de esquema de tabela .

Para exibir os arquivos que compõem uma partição do Amazon S3, escolha View partition (Visualizar partição). Para tabelas do Amazon S3, a coluna Key (Chave) exibe as teclas de partição usadas para particionar a tabela no datastore de origem. Particionar é uma maneira de separar uma tabela em partes relacionadas com base nos valores de uma coluna de chave, como data, local ou departamento. Para obter mais informações sobre partições, pesquise na Internet informações sobre "particionamento do Hive".

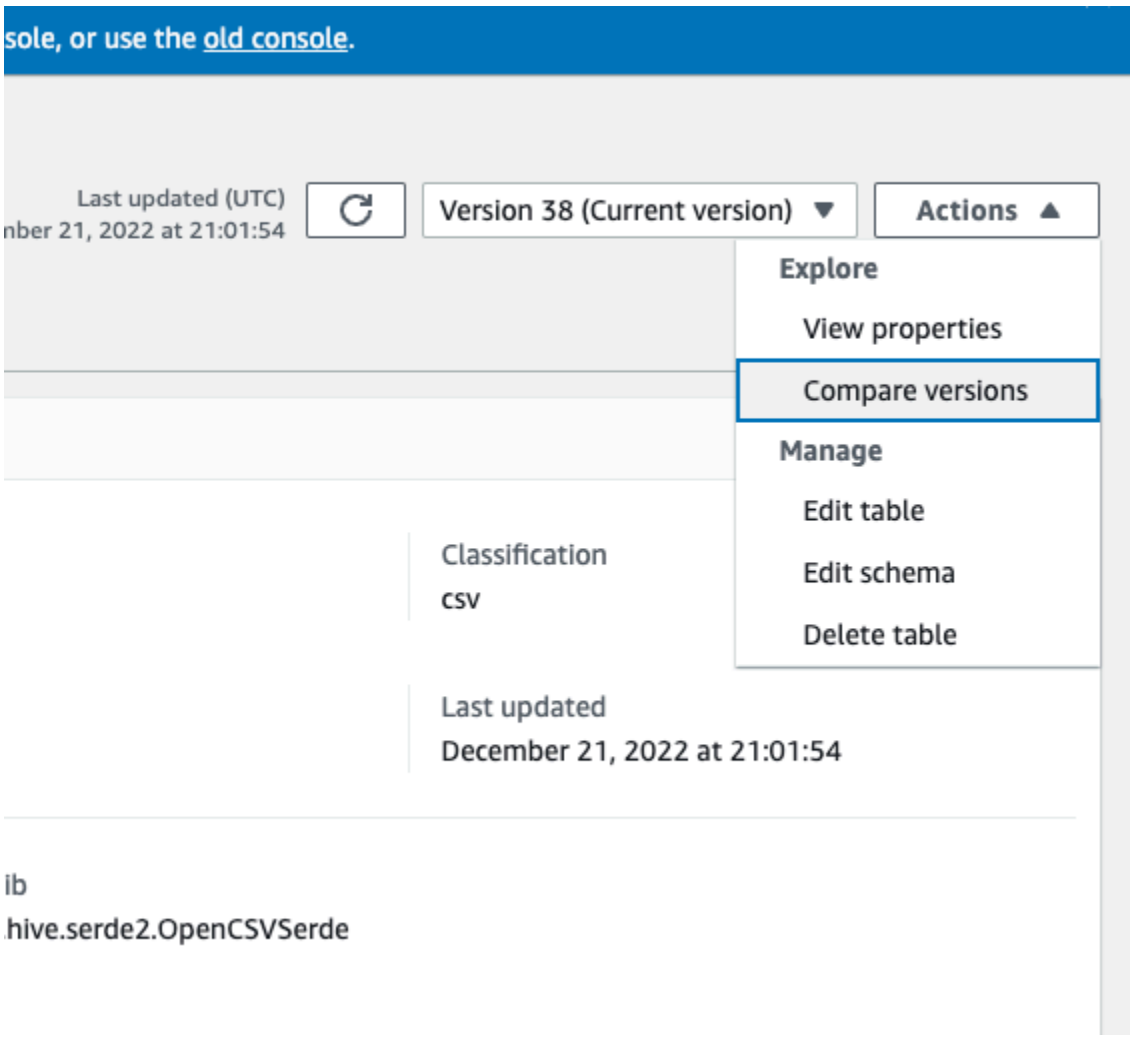
> **Note**
>
> Para obter orientação detalhada para visualizar os detalhes de uma tabela, consulte o tutorial Explore table no console.

Comparar versões de esquema de tabela

Ao comparar duas versões de esquema de tabela, você pode comparar alterações de linhas aninhadas expandindo-as e contraindo-as, comparar esquemas de duas versões lado a lado e visualizar as propriedades da tabela lado a lado.

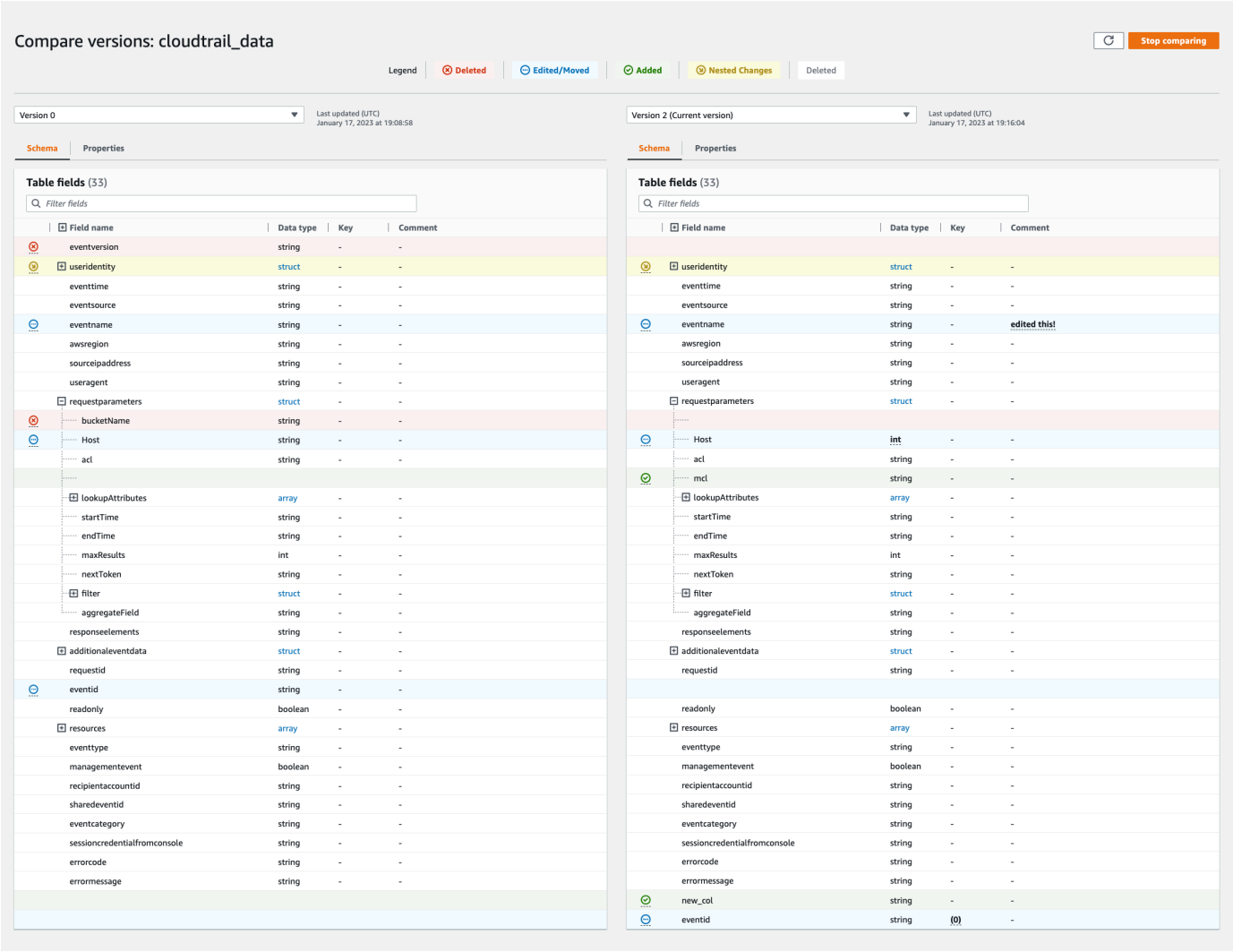
Para comparar versões

1. No console do AWS Glue, escolha Tabelas, depois Ações e Comparar versões.



2. Escolha uma versão para comparar no menu suspenso de versões. Ao comparar esquemas, a guia Esquema fica realçada em laranja.
3. Quando você compara tabelas entre duas versões, os esquemas de tabela são apresentados a você no lado esquerdo e no lado direito da tela. Isso permite que você determine visualmente as alterações comparando os campos de nome da coluna, tipo de dados, chave e comentários lado a lado. Quando há uma alteração, um ícone colorido exibe o tipo de alteração que foi feita.

- Excluído: exibido por um ícone vermelho indica onde a coluna foi removida de uma versão anterior do esquema da tabela.
- Editado ou movido: exibido por um ícone azul indica onde a coluna foi modificada ou movida em uma versão mais recente do esquema da tabela.
- Adicionado: exibido por um ícone verde indica onde a coluna foi adicionada a uma versão mais recente do esquema da tabela.
- Alterações aninhadas: exibido por um ícone amarelo indica onde a coluna aninhada contém as alterações. Escolha a coluna a ser expandida e visualize as colunas que foram excluídas, editadas, movidas ou adicionadas.



4. Use a barra de pesquisa de campos de filtro para exibir campos com base nos caracteres que você inserir aqui. Se você inserir um nome de coluna em qualquer uma das versões da tabela,

os campos filtrados serão exibidos nas duas versões da tabela para mostrar onde as alterações ocorreram.

5. Para comparar propriedades, escolha a guia Propriedades.
6. Para parar de comparar versões, escolha Interromper comparação para retornar à lista de tabelas.

Trabalhar com índices de partição no AWS Glue

Ao longo do tempo, centenas de milhares de partições são adicionadas a uma tabela. A API GetPartitions é usada para buscar as partições na tabela. A API retorna partições que correspondem à expressão fornecida na solicitação.

Vamos usar como exemplo uma tabela dados_vendas que é particionada pelas chaves País, Categoria, Ano, Mês e creationDate. Para obter os dados de vendas para todos os itens vendidos da categoria Livros no ano de 2020 após 15/8/2020, será necessário fazer uma solicitação `GetPartitions` com a expressão "Categoria = 'Livros' e creationDate > '2020-08-15'" ao Catálogo de Dados.

Se nenhum índice de partição estiver presente na tabela, o AWS Glue carrega todas as partições da tabela e, em seguida, filtra as partições carregadas usando a expressão de consulta fornecida pelo usuário na solicitação `GetPartitions`. A consulta leva mais tempo para ser executada à medida que o número de partições aumenta em uma tabela sem índices. Com um índice, a consulta `GetPartitions` tentará buscar um subconjunto das partições em vez de carregar todas as partições na tabela.

Tópicos

- Sobre índices de partição
- Criar uma tabela com índices de partição
- Adicionar um índice de partição a uma tabela existente
- Descrever índices de partição em uma tabela
- Limitações no uso de índices de partição
- Usar índices para uma chamada GetPartitions otimizada
- Integração com mecanismos

Sobre índices de partição

Ao criar um índice de partição, você especifica uma lista de chaves de partição que já existem em uma tabela específica. O índice de partição é uma sublista de chaves de partição definidas na tabela. Um índice de partição pode ser criado em qualquer permutação de chaves de partição definidas na tabela. Para a tabela dados_vendas acima os índices possíveis são (país, categoria, creationDate), (país, categoria, ano), (país, categoria), (país), (categoria, país, ano, mês), e assim por diante.

O Data Catalog concatenará os valores de partição na ordem fornecida no momento da criação do índice. O índice é criado consistentemente à medida que as partições são adicionadas à tabela. É possível criar índices para tipos de colunas string (string, char e varchar), numérico (int, bigint, long, tinyint e smallint) e data (aaaa-MM-dd).

Tipos de dados compatíveis

- Data: uma data no formato ISO, como `YYYY-MM-DD`. Por exemplo, data `2020-08-15`. O formato usa hífens (-) para separar o ano, o mês e o dia. O intervalo permitido para datas de indexação se estende de `0000-01-01` até `9999-12-31`.
- String: um literal de string entre aspas simples ou duplas.
- Char: dados de caractere de comprimento fixo, com um comprimento especificado entre 1 e 255, por exemplo, char(10).
- Varchar: dados de caracteres de comprimento variável com um tamanho especificado entre 1 e 65535, como varchar(10).
- Numérico: int, bigint, long, tinyint e smallint

Índices em dados dos tipos Numérico, String e Data são compatíveis com os operadores =, >, >=, <, <= e “between” (entre). Atualmente, a solução de indexação oferece suporte somente ao operador lógico `AND`. Subexpressões com os operadores “LIKE”, “IN”, “OR” e “NOT” são ignoradas na expressão para filtragem usando um índice. A filtragem da subexpressão ignorada é feita nas partições obtidas após a aplicação da filtragem de índice.

Para cada partição adicionada a uma tabela, há um item de índice correspondente criado. Para uma tabela com “n” partições , um índice de partição resultará em “n” itens de índice de partição. O índice de partição “m” na mesma tabela resultará em “m*n” itens de índice de partição. Cada item de índice de partição será cobrado de acordo com a política de preços AWS Glue para armazenamento no catálogo de dados. Para obter detalhes sobre o preço do objeto de armazenamento, consulte Preço do AWS Glue.

Criar uma tabela com índices de partição

Você pode criar um índice de partição durante a criação da tabela. O `CreateTable` solicita uma lista de objetos `PartitionIndex` como uma entrada. Um máximo de três índices de partição pode ser criado em uma determinada tabela. Cada índice de partição requer um nome e uma lista de `partitionKeys` definidos para a tabela. Os índices criados em uma tabela podem ser obtidos usando a API `GetPartitionIndexes`

Adicionar um índice de partição a uma tabela existente

Para adicionar um índice de partição a uma tabela existente, use a operação `CreatePartitionIndex`. Você só pode criar um `PartitionIndex` por operação `CreatePartitionIndex`. Adicionar um índice não afeta a disponibilidade de uma tabela, pois a tabela continua disponível enquanto os índices estão sendo criados.

O status do índice de uma partição adicionada é definido como CREATING (Criando) e a criação dos dados do índice é iniciada. Se o processo para criar os índices for bem-sucedido, indexStatus (status do índice) é atualizado para ACTIVE (Ativo) e, para um processo malsucedido, o status do índice é atualizado para FAILED (Falhou). A criação do índice pode falhar por vários motivos, e você pode usar a operação `GetPartitionIndexes` para recuperar os detalhes da falha. As possíveis falhas são:

- ENCRYPTED_PARTITION_ERROR (Erro de partição criptografada): a criação de índice em uma tabela com partições criptografadas não é suportada.
- INVALID_PARTITION_TYPE_DATA_ERROR (Erro de dados de tipo de partição inválidos): exibido quando o valor da `partitionKey` não é válido para o valor do tipo de dados da `partitionKey` correspondente. Por exemplo, uma `partitionKey` com o tipo de dados “int” tem um valor “foo”.
- MISSING_PARTITION_VALUE_ERROR (Erro de valor de partição ausente): exibido quando o `partitionValue` para uma `indexedKey` não está presente. Isso pode acontecer quando uma tabela não é particionada de forma consistente.
- UNSUPPORTED_PARTITION_CHARACTER_ERROR (Erro de caractere de partição não suportado): exibido quando o valor de uma chave de partição indexada contém os caracteres \u0000, \u0001 ou \u0002
- INTERNAL_ERROR (Erro interno): ocorreu um erro interno enquanto os índices eram criados.

Descrever índices de partição em uma tabela

Para buscar os índices de partição criados em uma tabela, use a operação `GetPartitionIndexes`. A resposta retorna todos os índices na tabela, juntamente com o status atual de cada índice (o `IndexStatus`).

O `IndexStatus` para um índice de partição será um dos seguintes:

- `CREATING`: o índice está sendo criado e ainda não está disponível para uso.
- `ACTIVE`: o índice está pronto para uso. As solicitações podem usar o índice para executar uma consulta otimizada.
- `DELETING`: o índice está sendo excluído e não pode mais ser usado. Um índice no estado ativo pode ser excluído usando a solicitação `DeletePartitionIndex`, que move o status de ACTIVE (Ativo) para DELETING (Excluindo).
- `FAILED`: falha na criação do índice em uma tabela existente. Cada tabela armazena os últimos dez índices com falha.

As possíveis transições de estado para índices criados em uma tabela existente são:

- CREATING → ACTIVE → DELETING
- CREATING → FAILED

Limitações no uso de índices de partição

Depois de criar um índice de partição, observe estas alterações na funcionalidade da tabela e da partição:

Criação de uma nova partição (após a adição do índice)

Depois que um índice de partição é criado em uma tabela, todas as novas partições adicionadas à tabela serão validadas para as verificações de tipo de dados para chaves indexadas. O valor da partição das chaves indexadas será validado para o formato do tipo de dados. Se a verificação do tipo de dados falhar, a operação de criação de partição falhará. Para a tabela dados_de_venda, se um índice for criado para chaves (categoria, ano) em que a categoria é do tipo `string` e ano do tipo `int`, a criação da nova partição com um valor de ANO como “foo” falhará.

Depois que os índices estiverem habilitados, a adição de partições com valores de chave indexados com os caracteres U+0000, U+00001 e U+0002 passará a falhar.

Atualizações de tabelas

Depois que um índice de partição é criado em uma tabela, você não pode modificar os nomes de chave de partição para chaves de partição existentes e não pode alterar o tipo, ou ordem, das chaves registradas com o índice.

Usar índices para uma chamada GetPartitions otimizada

Quando você chama `GetPartitions` em uma tabela com um índice, pode incluir uma expressão e, se aplicável, o Data Catalog usará um índice, se possível. A primeira chave do índice deve ser inserida na expressão para os índices a serem usados na filtragem. A otimização de índice na filtragem é aplicada como um melhor esforço. O Data Catalog tenta usar a otimização de índice tanto quanto possível, mas no caso de um índice ausente, ou operador não suportado, ele volta para a implantação existente de carregamento de todas as partições.

Para a tabela dados_de_venda anterior, vamos adicionar o índice [País, Categoria, Ano]. Se “País” não for inserido na expressão, o índice registrado não poderá filtrar partições usando índices. Você pode adicionar até três índices para suportar vários padrões de consulta.

Vamos pegar algumas expressões de exemplo e ver como os índices funcionam nelas:

Expressões	Como o índice será usado
País = 'EUA'	O índice será usado para filtrar partições.
País = 'EUA' and Categoria = 'Sapatos'	O índice será usado para filtrar partições.
Categoria = 'Sapatos'	Os índices não serão usados, pois “país” não foi fornecido na expressão. Todas as partições serão carregadas para retornar uma resposta.
País = 'EUA' and Categoria = 'Sapatos' and Ano > '2018'	O índice será usado para filtrar partições.
País = 'EUA' and Categoria = 'Sapatos' and Ano > '2018' and mês = 2	O índice será usado para buscar todas as partições com país = “EUA” e categoria = “sapatos” e ano > 2018. Em seguida, a filtragem com a expressão mês será realizada.

Expressões	Como o índice será usado
País = 'EUA' AND Categoria = 'Sapatos' OR Ano > '2018'	Os índices não serão usados, pois há um operador `OR` na expressão.
País = 'EUA' AND Categoria = 'Sapatos' AND (Ano = '2017' OR Ano = '2018')	O índice será usado para buscar todas as partições com país = “US” e categoria = “sapatos” e, em seguida, a filtragem com a expressão ano será realizada.
País in ('EUA', 'Reino Unido') AND Categoria = 'Sapatos'	Os índices não serão usados para filtrar, pois o operador `IN` não é suportado no momento.
País = 'EUA' AND Categoria in ('Sapatos', 'Livros')	O índice será usado para buscar todas as partições com país = “EUA”. A filtragem com a expressão Categoria será realizada em seguida.
País = 'EUA' AND Categoria in ('Sapatos', 'Livros') AND (creationDate > '2023-9-01')	O índice será usado para buscar todas as partições com país = "EUA" e creationDate > "2023-9-01". A filtragem com a expressão Categoria será realizada em seguida.

Integração com mecanismos

Redshift Spectrum, Amazon EMR e AWS Glue ETL Spark DataFrames são capazes de utilizar índices para buscar partições depois que os índices estiverem em um estado ACTIVE (Ativo) no AWS Glue. Athena e AWS Glue ETL Dynamic frames exigem que você siga etapas adicionais a fim de utilizar índices para melhorar consultas.

Habilitar filtragem de partições

Para ativar a filtragem de partições no Athena, é necessário atualizar as propriedades da tabela da seguinte forma:

1. No console do AWS Glue, escolha Tabelas em Catálogo de Dados.
2. Escolha uma tabela.
3. Em Ações, escolha Editar tabela.

4. Em Propriedades da tabela, adicione o seguinte:
   - Chave: `partition_filtering.enabled`
   - Valor: `true`
5. Escolha Aplicar.

Como alternativa, é possível definir esse parâmetro executando uma consulta ALTER TABLE SET PROPERTIES no Athena.

```
ALTER TABLE partition_index.table_with_index
SET TBLPROPERTIES ('partition_filtering.enabled' = 'true')
```

Integração a outros serviços da AWS

Embora você possa usar Crawler do AWS Glues para preencher o AWS Glue Data Catalog, há vários serviços da AWS que podem se integrar e preencher automaticamente o catálogo para você. As seções a seguir fornecem mais informações sobre os casos de uso específicos aceitos por serviços da AWS que podem preencher o Catálogo de dados.

Tópicos

- AWS Lake Formation
- Amazon Athena

AWS Lake Formation

O AWS Lake Formation é um serviço que facilita configurar um data lake seguro na AWS. O Lake Formation é construído sobre a plataforma AWS Glue, e tanto ele quanto o AWS Glue compartilham o mesmo AWS Glue Data Catalog. Você pode registrar sua localização de dados do Amazon S3 com o Lake Formation e usar o console do Lake Formation para criar bancos de dados e tabelas no Catálogo de Dados do AWS Glue, definir políticas de acesso aos dados e auditar o acesso aos dados em seu data lake desde um local central. Você pode usar o controle de acesso refinado do Lake Formation para gerenciar seus recursos existentes do catálogo de dados e os locais de dados do Amazon S3.

Com os dados registrados no Lake Formation, você pode compartilhar com segurança os recursos do Catálogo de Dados entre as entidades principais do IAM, contas da AWS, organizações da AWS e unidades organizacionais.

Para obter mais informações sobre a criação de recursos do Catálogo de Dados usando o Lake Formation, consulte Criar tabelas e bancos de dados do Catálogo de Dados no Guia do desenvolvedor do AWS Lake Formation.

Amazon Athena

O Amazon Athena usa o Catálogo de Dados para armazenar e recuperar metadados de tabela para os dados do Amazon S3 em sua conta da AWS. Os metadados da tabela permitem que o mecanismo de consulta do Athena saiba como localizar, ler e processar os dados que você deseja consultar.

Você pode preencher o AWS Glue Data Catalog usando diretamente as instruções `CREATE TABLE` do Athena. É possível definir e preencher manualmente os metadados do esquema e de partição no Catálogo de Dados sem precisar executar um crawler.

1. No console do Athena, crie um banco de dados que armazenará os metadados da tabela no Catálogo de Dados.
2. Use a instrução `CREATE EXTERNAL TABLE` para definir o esquema da sua fonte de dados.
3. Use a cláusula `PARTITIONED BY` para definir quaisquer chaves de partição se seus dados estiverem particionados.
4. Use a cláusula `LOCATION` para especificar o caminho do Amazon S3 em que seus arquivos de dados reais são armazenados.
5. Execute a instrução `CREATE TABLE`.

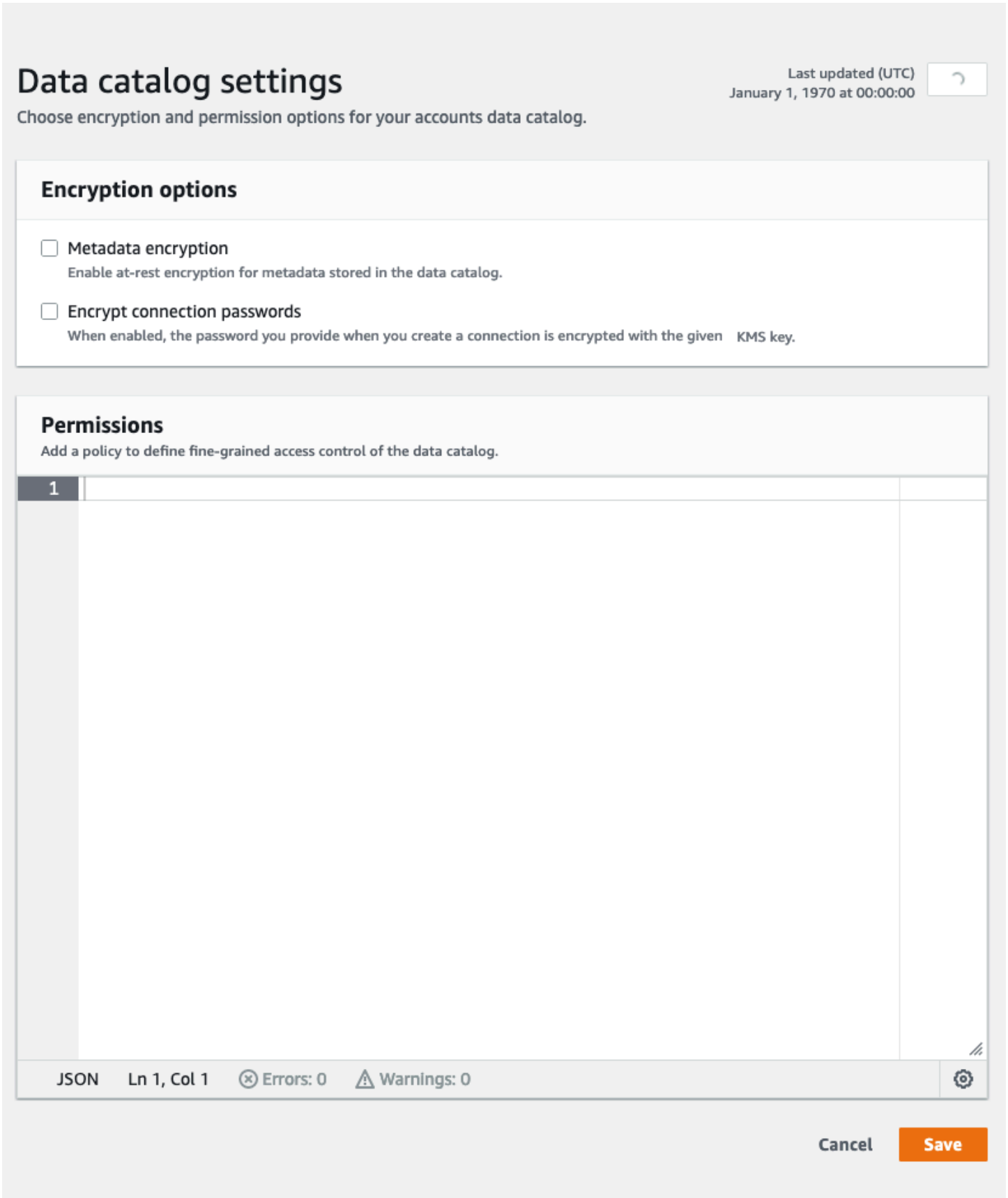
   Essa consulta cria os metadados da tabela no Catálogo de Dados com base no esquema e nas partições definidos, sem realmente fazer o crawling dos dados.

Você pode consultar a tabela no Athena e ela usará os metadados do Catálogo de Dados para acessar e consultar seus arquivos de dados no Amazon S3.

Para obter mais informações, consulte Criar bancos de dados e tabelas no Guia do usuário do Amazon Athena.

Configurações do Catálogo de Dados

A página de configurações do Data Catalog contém opções para definir opções de criptografia e permissões do Catálogo de Dados em sua conta.


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 alterar o controle de acesso granular do Data Catalog

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Escolha uma opção de criptografia.

   - Criptografia de metadados: marque essa caixa de seleção para criptografar os metadados no Data Catalog. Os metadados são criptografados em repouso usando a chave AWS Key Management Service (AWS KMS) que você especifica. Para ter mais informações, consulte Como criptografar seu Data Catalog.
   - Criptografar senhas de conexão: marque essa caixa de seleção para criptografar senhas no objeto de conexão do AWS Glue quando a conexão for criada ou atualizada. As senhas são criptografadas usando a AWS KMS chave que você especificar. Quando as senhas forem retornadas, elas são criptografadas. Esta opção é uma configuração global para todas as conexões do AWS Glue no Data Catalog. Se você desmarcar esta caixa de seleção, as senhas criptografadas anteriormente permanecerão criptografadas usando a chave que foi usada quando foram criadas ou atualizadas. Para obter mais informações sobre conexões do AWS Glue, consulte Conectar a dados.

     Ao habilitar essa opção, escolha uma AWS KMS chave ou escolha Insira uma chave ARN e forneça o nome de recurso da Amazon (ARN) para a chave. Insira o ARN no formato `arn:aws:kms:`*`region`*`:`*`account-id`*`:key/`*`key-id`* . Você também pode fornecer o ARN como um alias da chave, como `arn:aws:kms:`*`region`*`:`*`account-id`*`:alias/`*`alias-name`* .

     > **Important**
     >
     > Se esta opção estiver selecionada, qualquer usuário ou função que crie ou atualize uma conexão deve ter `kms:Encrypt` permissão na chave KMS especificada.

     Para ter mais informações, consulte Criptografar senhas de conexão.
3. Escolha Settings (Configurações) e, em seguida, no editor Permissions (Permissões), adicione a declaração de política para alterar o controle de acesso granular do Data Catalog de sua conta. Apenas uma política pode ser anexada a um Data Catalog de cada vez. Você pode colar uma política de recurso JSON nesse controle. Para ter mais informações, consulte Políticas baseadas em recursos no Glue AWS.

4. Escolha Save (Salvar) para atualizar o Data Catalog com as alterações feitas.

Você também pode usar AWS Glue operações da API para colocar, obter e excluir políticas de recursos.. Para ter mais informações, consulte APIs de segurança no AWS Glue.

Preencher e gerenciar tabelas transacionais

O Apache Iceberg, o Apache Hudi e o Linux Foundation Delta Lake são formatos de tabela de código aberto projetados para lidar com análises de dados em grande escala e workloads de data lake no Apache Spark.

Você pode preencher as tabelas do Iceberg, Hudi e Delta Lake no AWS Glue Data Catalog usando os seguintes métodos:

- Crawler do AWS Glue: os Crawler do AWS Glues podem descobrir e preencher automaticamente os metadados das tabelas do Iceberg, Hudi e Delta Lake no Catálogo de Dados. Para ter mais informações, consulte Usar crawlers para preencher o catálogo de dados .
- Trabalhos do AWS Glue ETL: é possível criar trabalhos de ETL para gravar dados nas tabelas do Iceberg, Hudi e Delta Lake e preencher seus metadados no Catálogo de Dados. Para obter mais informações, consulte Usar estruturas de data lake com trabalhos do AWS Glue ETL.
- Console do AWS Glue, console do AWS Lake Formation, AWS CLI ou API: você pode usar o console do AWS Glue, o console do Lake Formation ou a API da para criar e gerenciar definições de tabelas do Iceberg no Catálogo de Dados.

Tópicos

Criar tabelas no Apache Iceberg

Agora você pode criar tabelas do Apache Iceberg que usam o formato de dados do Apache Parquet no AWS Glue Data Catalog com dados residentes no Amazon S3. Uma tabela no catálogo de dados é a definição de metadados que representa os dados em um armazenamento de dados. Por padrão, o AWS Glue cria tabelas do Iceberg v2. Para saber a diferença entre as tabelas da v1 e v2, consulte Alterações de versão do formato na documentação do Apache Iceberg.

Apache Iceberg é um formato de tabela aberta para conjuntos de dados analíticos muito grandes. O Iceberg permite mudanças fáceis em seu esquema, também conhecido como evolução do esquema, o que significa que os usuários podem adicionar, renomear ou remover colunas de uma tabela de dados sem interromper os dados subjacentes. O Iceberg também fornece suporte para controle de versão de dados, o que permite que os usuários acompanhem as alterações nos dados ao longo do tempo. Isso ativa o atributo de viagem no tempo, que permite que os usuários acessem e consultem versões históricas dos dados e analisem as alterações nos dados entre atualizações e exclusões.

Você pode usar o AWS Glue, o console do Lake Formation ou a operação `CreateTable` na API do AWS Glue para criar uma tabela do Iceberg no Catálogo de Dados. Para obter mais informações, consulte a ação CreateTable (Python: create_table).

Ao criar uma tabela do Iceberg no catálogo de dados, você deve especificar o formato da tabela e o caminho do arquivo de metadados no Amazon S3 para poder realizar leituras e gravações.

Você pode usar o Lake Formation para proteger sua tabela do Iceberg usando permissões de controle de acesso refinadas ao registrar a localização de dados do Amazon S3 com o AWS Lake Formation. Para dados de origem no Amazon S3 e metadados que não estão registrados no Lake Formation, o acesso é determinado pelas políticas de permissões do IAM para ações do Amazon S3 e do AWS Glue. Para obter mais informações, consulte Gerenciar permissões.

> **Note**
>
> O catálogo de dados não oferece suporte à criação de partições e à adição de propriedades da tabela do Iceberg.

Pré-requisitos

Para criar tabelas Iceberg no catálogo de dados e configurar as permissões de acesso aos dados do Lake Formation, você precisa preencher os seguintes requisitos:

1. Permissões necessárias para criar tabelas do Iceberg sem os dados registrados no Lake Formation.

   Além das permissões necessárias para criar uma tabela no catálogo de dados, o criador da tabela precisa as seguintes permissões:

   - `s3:PutObject` no recurso arn:aws:s3:::{bucketName}
   - `s3:GetObject` no recurso arn:aws:s3:::{bucketName}

- `s3:DeleteObject` no recurso arn:aws:s3:::{bucketName}

2. Permissões necessárias para criar tabelas do Iceberg com dados registrados no Lake Formation:

   Para usar o Lake Formation para gerenciar e proteger os dados em seu data lake, registre sua localização no Amazon S3 que tenha os dados para tabelas com o Lake Formation. Isso é para que a Lake Formation possa fornecer credenciais para serviços analíticos AWS como Athena, Redshift Spectrum e Amazon EMR para acessar dados. Para obter mais informações sobre o registro de um local do Amazon S3, consulte Adicionar um local do Amazon S3 ao seu data lake.

   Uma entidade principal que lê e grava os dados subjacentes registrados no Lake Formation exige as seguintes permissões:

   - `lakeformation:GetDataAccess`
   - `DATA_LOCATION_ACCESS`

     Uma entidade principal que tem permissões de localização de dados em um local também tem permissões de localização em todos os locais secundários.

     Para obter mais informações sobre permissões de localização de dados, consulte Controle de acesso a dados subjacente.

Para permitir a compactação, o serviço precisa assumir um perfil do IAM que tenha permissões para atualizar tabelas no catálogo de dados. Para obter detalhes, consulte Pré-requisitos de otimização de tabelas

Criar uma tabela no Iceberg

Você pode criar tabelas do Iceberg v1 e v2 usando o AWS Glue, o console do Lake Formation ou a AWS Command Line Interface conforme documentado nesta página. Você também pode criar tabelas do Iceberg usando o Crawler do AWS Glue. Para obter mais informações, consulte Catálogo de dados e crawlers no Guia do desenvolvedor do AWS Glue.

Para criar uma tabela no Iceberg

Console

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.

2. Em catálogo de dados, escolha Tabelas e use o botão Criar tabela para especificar os seguintes atributos:
   - Nome da tabela: insira um nome para a tabela. Se você estiver usando o Athena para acessar tabelas, use essas dicas de nomenclatura no Guia do usuário do Amazon Athena.
   - Banco de dados: escolha um banco de dados existente ou crie um novo.
   - Descrição: a descrição da tabela. Você pode escrever uma descrição para ajudá-lo a entender o conteúdo da tabela.
   - Formato da tabela: para Formato da tabela, escolha Apache Iceberg.
   - Ativar compactação: escolha Ativar compactação para compactar objetos pequenos do Amazon S3 na tabela em objetos maiores.
   - Perfil do IAM: para executar a compactação, o serviço assume um perfil do IAM em seu nome. Você pode escolher um perfil do IAM usando o menu suspenso. Certifique-se de que a função tenha as permissões necessárias para habilitar a compactação.

     Para saber mais sobre as permissões necessárias, consulte Pré-requisitos de otimização de tabelas .
   - Localização: especifique o caminho para a pasta no Amazon S3 que armazena a tabela de metadados. O Iceberg precisa de um arquivo de metadados e de um local no catálogo de dados para poder realizar leituras e gravações.
   - Esquema: escolha Adicionar colunas para adicionar colunas e tipos de dados das colunas. Você tem a opção de criar uma tabela vazia e atualizar o esquema posteriormente.
     O catálogo de dados oferece suporte aos tipos de dados do Hive. Para obter mais informações, consulte Tipos de dados do Hive.

     O Iceberg permite que você evolua o esquema e a partição depois de criar a tabela. Você pode usar as consultas do Athena para atualizar o esquema da tabela e as consultas do Spark para atualizar as partições.

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/"
            }
        }'
```

Otimizar tabelas Iceberg

Os data lakes do Amazon S3 usando formatos de tabela aberta, como o Apache Iceberg, armazenam os dados como objetos do Amazon S3. Ter milhares de pequenos objetos Amazon S3 em uma tabela de data lake aumenta a sobrecarga de metadados nas tabelas Iceberg e afeta o desempenho de leitura. Para melhor desempenho de leitura por serviços de análise da AWS, como o Amazon Athena e o Amazon EMR, e trabalhos de ETL do AWS Glue, o AWS Glue Data Catalog fornece compactação gerenciada (um processo que compacta pequenos objetos do Amazon S3 em objetos maiores) para tabelas Iceberg no catálogo de dados. Você pode usar o console do Lake Formation, o console do AWS Glue, a AWS CLI ou a API da AWS para ativar ou desativar a compactação de tabelas individuais do Iceberg que estão no catálogo de dados.

O otimizador de tabelas monitora constantemente as partições da tabela e inicia o processo de compactação quando o limite é excedido para o número de arquivos e tamanhos de arquivo. Uma tabela do Iceberg se qualificará para compactação se o tamanho de arquivo especificado na propriedade write.target-file-size-bytes estiver dentro do intervalo de 128 MB a 512 MB. No Catálogo de Dados, o processo de compactação começará se a tabela tiver mais de cinco arquivos, cada um menor que 75% da propriedade write.target-file-size-bytes.

Por exemplo, você tem uma tabela com o limite de tamanho de arquivo definido como 512 MB na propriedade write.target-file-size-bytes (dentro do intervalo prescrito de 128 MB a 512 MB), e a tabela contém 10 arquivos. Se 6 dos 10 arquivos tiverem menos de 384 MB (0,75 x 512) cada, o Catálogo de Dados acionará a compactação.

O catálogo de dados executa a compactação sem interferir nas consultas simultâneas. O catálogo de dados oferece suporte à compactação de dados somente para tabelas no formato Parquet.

Para conhecer tipos de dados, formatos de compactação e limitações compatíveis, consulte Formatos e limitações compatíveis para compactação gerenciada de dados .

Tópicos

Pré-requisitos de otimização de tabelas

O otimizador de tabela assume as permissões do perfil do (IAM) AWS Identity and Access Management que você especificou ao habilitar a compactação para uma tabela. O perfil do IAM deve ter as permissões para ler dados e atualizar metadados no catálogo de dados. Você pode criar um perfil do IAM e anexar as políticas em linha a seguir:

- Adicione a seguinte política em linha que concede ao Amazon S3 permissões de leitura/gravação no local para dados que não estão registrados no Lake Formation. Essa política também inclui permissões para atualizar a tabela no catálogo de dados e permitir que o AWS Glue adicione logs em logs do Amazon CloudWatch e a publicação de métricas. Para dados de origem no Amazon S3 que não estão registrados no Lake Formation, o acesso é determinado pelas políticas de permissões do IAM para o Amazon S3 e ações AWS Glue.

  Nas políticas em linha a seguir, substitua `bucket-name` pelo nome do bucket do Amazon S3, `aws-account-id` e `region` por um número de conta da AWS e por uma região do catálogo de dados válidos, `database_name` pelo nome do seu banco de dados e `table_name` pelo nome da tabela.

```
{
    "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:*"
        }
    ]
```

```
}
```

- Use a política a seguir para habilitar a compactação de dados registrados no Lake Formation.

  Se o perfil de compactação não tiver permissões de grupo `IAM_ALLOWED_PRINCIPALS` concedidas na tabela, o perfil exigirá as permissões `ALTER`, `DESCRIBE`, `INSERT` e `DELETE` do Lake Formation na tabela.

  Para obter mais informações sobre o registro de um bucket do Amazon S3 com o Lake Formation, consulte Adicionar um local do Amazon S3 ao seu data lake.

```
{
 "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 tabelas Iceberg com dados em buckets do Amazon S3 criptografados usando criptografia do lado do servidor, a função de compactação exige permissões para descriptografar objetos do Amazon S3 e gerar uma nova chave de dados para gravar objetos nos buckets criptografados. Adicione a seguinte política à chave do AWS KMS desejada. Oferecemos suporte somente à criptografia em nível de bucket.

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::<aws-account-id>:role/<compaction-role-name>"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

- (Opcional) Para locais de dados registrados no Lake Formation, o perfil usado para registrar a localização exige permissões para descriptografar objetos do Amazon S3 e gerar uma nova chave de dados para gravar objetos nos buckets criptografados. Para obter mais informações, consulte Registrar um local do Amazon S3.
- (Opcional) Se a chave do AWS KMS estiver armazenada em uma conta da AWS diferente, você precisará incluir as permissões a seguir no perfil de compactação.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
```

```
            "Resource": ["arn:aws:kms:<REGION>:<KEY_OWNER_ACCOUNT_ID>:key/<KEY_ID>" ]
        }
    ]
}
```

- A função que você usa para executar a compactação deve ter a permissão `iam:PassRole` da função.

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

- Adicione a política de confiança a seguir à função para que o serviço AWS Glue assuma o perfil do IAM para executar o processo de compactação.

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

Permitir compactação

Você pode usar o console do Lake Formation, o console do AWS Glue, a AWS CLI ou a API da AWS para ativar ou desativar a compactação de tabelas individuais do Iceberg que estão no catálogo de dados. Para novas tabelas, você pode escolher o Apache Iceberg como formato de tabela e ativar a compactação ao criar a tabela. A compactação está desabilitada por padrão para novas tabelas.

Console

Para habilitar compactação

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/ e faça login como administrador do data lake, criador da tabela ou um usuário que tenha recebido as permissões `glue:UpdateTable` e `lakeformation:GetDataAccess` na tabela.
2. No painel de navegação, em Catálogo de dados, escolha Tabelas.
3. Na página Tabelas, escolha uma tabela em formato de tabela aberta para a qual você deseja habilitar a compactação e, em seguida, no menu Ações, escolha Habilitar compactação.
4. Você também pode ativar a compactação selecionando a tabela e abrindo a página de Detalhes da tabela. Escolha a guia Otimização da tabela na seção inferior da página e escolha Ativar compactação.
5. Em seguida, selecione um perfil do IAM existente no menu suspenso com as permissões mostradas na seção Pré-requisitos de otimização de tabelas .

   Quando você escolhe a opção Criar perfil do IAM, o serviço cria um perfil personalizado com as permissões necessárias para realizar a compactação.

   Siga as etapas abaixo para atualizar um perfil do IAM existente:

   a. Para atualizar a política de permissões para o perfil do IAM, no console do IAM, acesse a função do IAM que está sendo usada para executar a compactação.
   b. Na seção Adicionar permissões, escolha Criar política. Na janela recém-aberta do navegador, crie uma nova política para usar com sua função.
   c. Na página Criar política, escolha a guia JSON. Copie o código JSON mostrado nos Pré-requisitos no campo do editor de políticas.

AWS CLI

O exemplo a seguir mostra como habilitar a compactação. Substitua o ID da conta por um ID de conta da AWS válido. Substitua o nome do banco de dados e o nome da tabela pelo nome real da tabela do Iceberg e pelo nome do banco de dados. Substitua o `roleArn` pelo nome do recurso (ARN) AWS do perfil do IAM e o nome do perfil do IAM que tem as permissões necessárias para executar a compactação.

```
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

Chame a operação `CreateTableOptimizer` para ativar a compactação de uma tabela.

Depois de ativar a compactação, a guia Otimização de tabela mostra os seguintes detalhes da compactação (após aproximadamente 15 a 20 minutos):

Horário de início

A época em que o processo de compactação começou no Lake Formation. O valor é um timestamp no horário UTC.

Horário de término

A hora em que o processo de compactação terminou no catálogo de dados. O valor é um timestamp no horário UTC.

Status

O status de execução da compactação. Os valores são sucesso ou falha.

Arquivos compactados

Número total de arquivos compactados.

Bytes compactados

Número total de bytes compactados.

Desabilitar compactação

Você pode desativar a compactação automática para uma tabela específica do Apache Iceberg usando o console do AWS Glue ou a AWS CLI.

Console

1. Escolha catálogo de dados e escolha Tabelas. Na lista de tabelas, escolha a tabela em formato de tabela aberta que você deseja desativar a compactação.
2. Você pode escolher uma tabela Iceberg e escolher Desativar compactação em Ações.

   Você também pode desabilitar a compactação da tabela escolhendo Desabilitar compactação na seção inferior da página Detalhes das tabelas.
3. Escolha Desabilitar compactação na mensagem de confirmação. Você poderá reabilitar a compactação mais tarde.

   Após a confirmação, a compactação é desativada e o status de compactação da tabela volta para `Off`.

AWS CLI

No exemplo a seguir, substitua o ID da conta por um ID de conta da AWS válido. Substitua o nome do banco de dados e o nome da tabela pelo nome real da tabela do Iceberg e pelo nome do banco de dados. Substitua o `roleArn` pelo nome do recurso (ARN) AWS do perfil do IAM e o nome real do perfil do IAM que tem as permissões necessárias para executar a compactação.

```
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

Chame a operação UpdateTableOptimizer para desativar a compactação de uma tabela específica.

Visualizar detalhes de compactação

Você pode visualizar o status de compactação do Apache Iceberg no console do Lake Formation, na AWS CLI ou usando as operações da API da AWS.

Console

Para visualizar o status de compactação das tabelas do Iceberg (console)

- Você pode visualizar o status de compactação das tabelas Iceberg no console do Lake Formation escolhendo Tabelas em catálogo de dados. O campo Status da compactação mostra o status da execução da compactação. É possível exibir o formato da tabela e o status da compactação usando as preferências da tabela.
- Para ver o histórico de execução de compactação de uma tabela específica, escolha Tabelas em AWS Glue Data Catalog e escolha uma tabela para visualizar seus detalhes. A guia Otimização da tabela mostra o histórico de compactação da tabela.

AWS CLI

É possível visualizar os detalhes da compactação usando a AWS CLI.

Nos exemplos a seguir, substitua o ID da conta por um ID de conta da AWS válido, o nome do banco de dados e o nome da tabela pelo nome real da tabela Iceberg.

- Para obter os detalhes da última execução de compactação para uma tabela

```
aws get-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

- Use o exemplo a seguir para recuperar o histórico de um otimizador para uma tabela específica.

```
aws list-table-optimizer-runs \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

- O exemplo a seguir mostra como recuperar a execução de compactação e os detalhes de configuração de vários otimizadores. Você pode especificar no máximo 20 otimizadores.

```
aws glue batch-get-table-optimizer \
--entries '[{"catalogId":"123456789012", "databaseName":"iceberg_db",
 "tableName":"iceberg_table", "type":"compaction"}]'
```

AWS API

- Use a operação `GetTableOptimizer` para recuperar os detalhes da última execução de um otimizador.
- Use a operação `ListTableOptimizerRuns` para recuperar o histórico de um determinado otimizador em uma tabela específica. Você pode especificar 20 otimizadores em uma única chamada de API.
- Use a operação `BatchGetTableOptimizer` para recuperar detalhes de configuração de vários otimizadores em sua conta. Esta operação não oferece suporte a chamadas entre contas.

Visualizar métricas do Amazon CloudWatch

Depois de executar a compactação com sucesso, o serviço cria métricas Amazon CloudWatch sobre o desempenho do trabalho de compactação. Você pode acessar as Métricas do CloudWatch e escolher Métricas, Todas as métricas. Você pode filtrar métricas pelo namespace específico (por exemplo, do AWS Glue), nome da tabela ou nome do banco de dados.

Para obter mais informações, consulte Visualizar métricas disponíveis no Guia do usuário do Amazon CloudWatch.

- Número de bytes compactados
- Número de arquivos compactados

- Número de DPU alocado para o trabalho
- Duração do trabalho (horas)

Excluindo um otimizador

Você pode excluir um otimizador e os metadados associados à tabela usando a AWS CLI ou a operação de API da AWS.

Execute o comando AWS CLI a seguir para excluir o histórico de compactação de uma tabela.

```
aws glue delete-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

Use a operação `DeleteTableOptimizer` para excluir um otimizador para uma tabela.

Formatos e limitações compatíveis para compactação gerenciada de dados

Para melhor desempenho de leitura por serviços de AWS análise, como Amazon Athena, Amazon EMR e trabalhos de AWS Glue ETL, AWS Glue Data Catalog fornece compactação gerenciada (um processo que compacta pequenos objetos do Amazon S3 em objetos maiores) para tabelas Iceberg no Data Catalog.

A compactação de dados aceita uma variedade de tipos de dados e formatos de compactação para leitura e gravação de dados, incluindo a leitura de dados de tabelas criptografadas.

A compactação de dados suporta:

- Tipos de arquivo: Parquet
- Tipos de dados: Booleano, Inteiro, Longo, Flutuante, Duplo, String, Decimal, Data, Hora, Timestamp, String, UUID, Binário
- Compactação: zstd, gzip, snappy, não compactado
- Criptografia: a compactação de dados oferece suporte somente à criptografia padrão do Amazon S3 (SSE-S3) e a criptografia do KMS no lado do servidor (SSE-KMS).
- Compactação do compartimento
- Evolução do esquema

- Tabelas com tamanho de arquivo de destino (gravação). target-file-size-bytes propriedade na configuração do iceberg) dentro da faixa inclusiva de 128 MB a 512 MB.
- Regiões
  - Ásia-Pacífico (Tóquio)
  - Ásia-Pacífico (Seul)
  - Ásia-Pacífico (Mumbai)
  - Ásia-Pacífico (Singapura)
  - Europa (Irlanda)
  - Europa (Londres)
  - Europa (Frankfurt)
  - Leste dos EUA (Norte da Virgínia)
  - Leste dos EUA (Ohio)
  - Oeste dos EUA (N. da Califórnia)
  - América do Sul (São Paulo)
- Você pode executar a compactação a partir da conta em que o catálogo de dados reside quando o bucket do Amazon S3 que armazena os dados subjacentes estiver em outra conta. Para fazer isso, a função de compactação exige acesso ao bucket do Amazon S3.

Atualmente, a compactação de dados não oferece suporte a:

- Tipos de arquivo: Avro, ORC
- Tipos de dados: Fixo
- Compactação: brotli, lz4
- Compactação de arquivos enquanto a especificação da partição evolui.
- Classificação regular ou classificação por ordem z
- Mesclar ou excluir arquivos: o processo de compactação ignora os arquivos de dados que têm arquivos excluídos associados a eles.
- Compactação em tabelas de contas cruzadas: não é possível executar a compactação em tabelas de contas cruzadas.
- Compactação de tabelas entre regiões: não é possível executar a compactação de tabelas entre regiões.
- Habilitando a compactação em links de recursos

- Endpoints da VPC para buckets do Amazon S3
- Gerenciador de bloqueio do DynamoDB — Ao usar a compactação de dados, nenhum outro trabalho de carregamento de dados deve ser usado como org.apache.iceberg.aws.dynamodb. `lock-impl` DynamoDbLockManager.

Gerenciar o Catálogo de Dados

O AWS Glue Data Catalog é um repositório central de metadados que armazena metadados estruturais e operacionais para seus conjuntos de dados do Amazon S3. Gerenciar o Catálogo de Dados de forma eficaz é crucial para manter a qualidade, o desempenho, a segurança e a governança dos dados.

Ao entender e aplicar essas práticas de gerenciamento de Catálogos de Dados, é possível garantir que seus metadados permaneçam precisos, eficientes, seguros e bem governados à medida que seu cenário de dados evolui.

Esta seção aborda os seguintes aspectos do gerenciamento do Catálogo de Dados:

- Atualização do esquema e das partições da tabela Conforme seus dados evoluírem, talvez seja necessário atualizar o esquema da tabela ou a estrutura de partições definida no Catálogo de Dados. Para obter mais informações sobre como fazer essas atualizações programaticamente usando o AWS Glue ETL, consulte Atualizar esquemas e adicionar novas partições ao Catálogo de Dados em trabalhos do AWS Glue ETL.
- Gerenciar estatísticas de colunas: estatísticas de colunas precisas ajudam a otimizar os planos de consulta e melhorar a performance. Para obter mais informações sobre como gerar, atualizar e gerenciar estatísticas de colunas, consulte Otimizar a performance da consulta usando estatísticas de coluna.
- Criptografar o Catálogo de Dados Para proteger metadados confidenciais, é possível criptografar seu Catálogo de Dados usando o AWS Key Management Service (AWS KMS). Esta seção explica como habilitar e gerenciar a criptografia em seu Catálogo de Dados.
- Proteger o Catálogo de Dados com o AWS Lake Formation O Lake Formation fornece uma abordagem abrangente para a segurança e o controle de acesso do data lake. É possível usar o Lake Formation para proteger e controlar o acesso ao seu catálogo de dados e aos dados subjacentes.

Tópicos

- Atualizar esquemas e adicionar novas partições ao Catálogo de Dados em trabalhos do AWS Glue ETL
- Otimizar a performance da consulta usando estatísticas de coluna
- Como criptografar seu Data Catalog
- Proteger seu catálogo de dados usando o Lake Formation

Atualizar esquemas e adicionar novas partições ao Catálogo de Dados em trabalhos do AWS Glue ETL

O trabalho de extração, transformação e carregamento (ETL) pode criar partições de tabelas no armazenamento de dados de destino. Os esquemas do conjunto de dados podem evoluir e divergir do esquema do esquema do AWS Glue Data Catalog ao longo do tempo. Os trabalhos de ETL do AWS Glue agora oferecem vários recursos que podem ser usados no script de ETL para atualizar os esquemas e as partições no Data Catalog. Esses recursos permitem visualizar os resultados do trabalho de ETL no Data Catalog sem precisar executar o crawler novamente.

Novas partições

Se quiser visualizar as novas partições no AWS Glue Data Catalog, é possível realizar um dos seguintes procedimentos:

- Quando o trabalho terminar, execute novamente o crawler e exiba as novas partições no console quando o crawler terminar.
- Quando o trabalho terminar, exiba as novas partições no console imediatamente, sem precisar executar novamente o crawler. Você pode habilitar esse recurso adicionando algumas linhas de código ao seu script de ETL, como exibido nos exemplos a seguir. O código usa o argumento `enableUpdateCatalog` para indicar que o Data Catalog deve ser atualizado durante a execução do trabalho à medida que as novas partições são criadas.

Método 1

Passe `enableUpdateCatalog` e `partitionKeys` em um argumento de opções.

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

Passe `enableUpdateCatalog` e `partitionKeys` em `getSink()` e chame o objeto `setCatalogInfo()` no `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)
```

Agora, é possível criar novas tabelas de catálogo, atualizar tabelas existentes com esquema modificado e adicionar novas partições de tabelas no Data Catalog usando um trabalho de ETL do AWS Glue, sem a necessidade de executar crawlers novamente.

Atualizar esquema da tabela

Se você quiser substituir o esquema da tabela do Data Catalog, é possível realizar um dos seguintes procedimentos:

- Quando o trabalho for concluído, execute o crawler novamente e verifique se está configurado para atualizar a definição da tabela também. Visualize as partições no console junto com atualizações do esquema, quando o crawler finalizar. Para obter mais informações, consulte Configurar um crawler com a API.
- Quando o trabalho for concluído, visualize o esquema modificado no console imediatamente, sem precisar executar o crawler novamente. Você pode habilitar esse recurso adicionando algumas linhas de código ao seu script de ETL, como exibido nos exemplos a seguir. O código usa `enableUpdateCatalog` definido como true e `updateBehavior` definido como `UPDATE_IN_DATABASE`, o que indica substituir o esquema e adicionar novas partições no Data Catalog durante a execução do trabalho.

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)
```

Também é possível definir o valor de `updateBehavior` como `LOG` se quiser impedir que o esquema da tabela seja substituído, mas ainda quiser adicionar novas partições. O valor padrão de `updateBehavior` é `UPDATE_IN_DATABASE`, portanto, se você não defini-lo explicitamente, o esquema da tabela será substituído.

Se `enableUpdateCatalog` não estiver definido como true, independentemente da opção selecionada para `updateBehavior`, o trabalho de ETL não atualizará a tabela no Data Catalog.

Criar novas tabelas

Também é possível usar as mesmas opções para criar uma tabela no Data Catalog. Também é possível especificar o banco de dados e o nome da nova tabela usando `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)
```

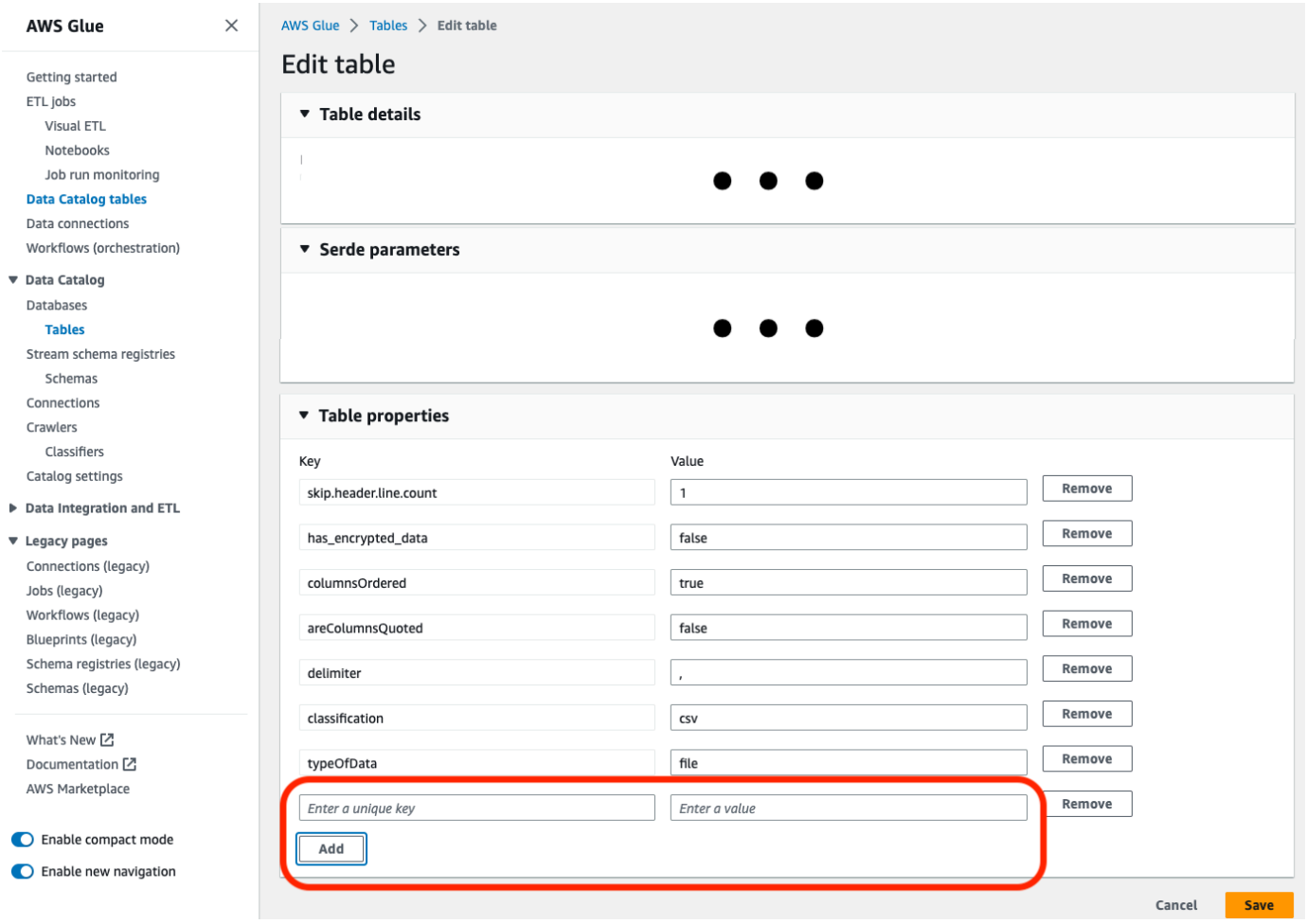
Scala

```
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)
```

Restrições

Observe as seguintes restrições:

- Somente os destinos do Amazon Simple Storage Service (Amazon S3) são suportados.
- O atributo `enableUpdateCatalog` é suportado para tabelas governadas.
- Somente os seguintes formatos são compatíveis: `json`, `csv`, `avro` e `parquet`.
- Para criar ou atualizar tabelas com a classificação `parquet`, você deve utilizar o gravador de parquet do AWS Glue otimizado para DynamicFrames. Isso pode ser feito de uma das seguintes maneiras:
  - se você estiver atualizando uma tabela existente no catálogo com classificação `parquet`, a tabela deve ter a propriedade de tabela `"useGlueParquetWriter"` definida como `true` antes de você atualizá-la. Você pode definir essa propriedade por meio de APIs/SDK do AWS Glue, por meio do console ou por meio de uma instrução DDL do Athena.



Depois que a propriedade da tabela do catálogo estiver definida, você poderá usar o seguinte trecho de código para atualizar a tabela do catálogo com os novos dados:

```
glueContext.write_dynamic_frame.from_catalog(
    frame=frameToWrite,
    database="dbName",
    table_name="tableName",
    additional_options={
        "enableUpdateCatalog": True,
        "updateBehavior": "UPDATE_IN_DATABASE"
    }
)
```

- Se a tabela ainda não existir no catálogo, você pode utilizar o método `getSink()` em seu script com `connection_type="s3"` para adicionar a tabela e suas partições ao catálogo, além de gravar os dados no Amazon S3. Forneça as `partitionKeys` e a `compression` para seu fluxo de trabalho.

```
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)
```

- O valor de formato `glueparquet` é um método herdado para habilitar o gravador parquet AWS Glue.

- Quando `updateBehavior` for definido como `LOG`, as novas partições serão adicionadas somente se o esquema `DynamicFrame` for equivalente a, ou contiver, um subconjunto das colunas definidas no esquema da tabela do Data Catalog.

- As atualizações de esquema não são aceitas para tabelas não particionadas (que não usam a opção “partitionKeys”).
- As partitionKeys devem ser equivalentes e estar na mesma ordem, entre o parâmetro informado no script de ETL e as partitionKeys no esquema da tabela do Data Catalog.
- Atualmente, esse recurso ainda não suporta atualização/criação de tabelas nas quais os esquemas de atualização são aninhados (por exemplo, matrizes dentro de estruturas).

Para ter mais informações, consulte the section called “AWS Glue para Spark”.

Trabalhar com conexões MongoDB em trabalhos de ETL

Você pode criar uma conexão para MongoDB e, em seguida, usar essa conexão em seu trabalho do AWS Glue. Para obter mais informações, consulte the section called “Conexões do MongoDB” no guia de programação do AWS Glue. O `url`, `username` e `password` da conexão são armazenados na conexão do MongoDB. Outras opções podem ser especificadas em seu script de trabalho de ETL usando o parâmetro `additionalOptions` do `glueContext.getCatalogSource`. As outras opções podem incluir:

- `database`: (obrigatório) o banco de dados MongoDB do qual fazer a leitura.
- `collection`: (obrigatório) a coleção do MongoDB da qual fazer a leitura.

Ao colocar as informações de `database` e `collection` dentro do script de trabalho de ETL, você pode usar a mesma conexão em vários trabalhos.

1. Criar uma conexão do AWS Glue Data Catalog para a fonte de dados do MongoDB. Consulte "connectionType": "mongodb" para uma descrição dos parâmetros de conexão. É possível criar a conexão ao utilizar o console, as APIs ou a CLI.
2. Crie um banco de dados no AWS Glue Data Catalog para armazenar as definições de tabela para seus dados do MongoDB. Consulte Criar bancos de dados Para mais informações.
3. Crie um crawler que rastreie os dados no MongoDB usando as informações na conexão para se conectar ao MongoDB. O crawler cria as tabelas no AWS Glue Data Catalog que descrevem as tabelas no banco de dados MongoDB que você usa em seu trabalho. Consulte Usar crawlers para preencher o catálogo de dados Para mais informações.
4. Crie um trabalho com um script personalizado. É possível criar o trabalho usando o console, as APIs ou a CLI. Para obter mais informações, consulte Adição de trabalhos no AWS Glue.

5. Escolha os destinos de dados para o seu trabalho. As tabelas que representam o destino dos dados podem ser definidas no Data Catalog, ou seu trabalho pode criar as tabelas de destino quando for executado. Você escolhe um local de destino ao criar o trabalho. Se o destino exigir uma conexão, ela também será referenciada no seu trabalho. Se o trabalho precisar de vários destinos de dados, você poderá adicioná-los posteriormente editando o script.
6. Personalize o ambiente de processamento de trabalhos informando os argumentos para seu trabalho e o script gerado.

   A seguir, encontra-se um exemplo de como criar um `DynamicFrame` do banco de dados MongoDB com base na estrutura da tabela definida no Data Catalog. O código usa `additionalOptions` para fornecer as informações adicionais sobre a origem dos dados:

   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. Execute o trabalho, sob demanda ou por meio de um acionador.

Otimizar a performance da consulta usando estatísticas de coluna

Você pode calcular estatísticas em nível de coluna para tabelas do AWS Glue Data Catalog em formatos de dados como Parquet, ORC, JSON, ION, CSV e XML sem precisar configurar pipelines de dados adicionais. As estatísticas de colunas ajudam você a entender os perfis de dados obtendo insights sobre os valores em uma coluna. O Catálogo de Dados possibilita a geração de estatísticas para valores de colunas, como valor mínimo, valor máximo, valores nulos totais, valores distintos totais, comprimento médio dos valores e ocorrências totais de valores reais.

Os serviços analíticos da AWS, como o Amazon Redshift e o Amazon Athena, podem usar essas estatísticas de colunas para gerar planos de execução de consultas e escolher o plano ideal que melhore o desempenho da consulta.

Você pode configurar para executar a tarefa de geração de estatísticas de coluna usando o console do AWS Glue ou a AWS CLI. Quando você inicia o processo, o AWS Glue inicia um trabalho do Spark em segundo plano e atualiza os metadados da tabela AWS Glue no Catálogo de Dados. Você pode visualizar as estatísticas da coluna usando o console do AWS Glue ou a AWS CLI ou chamando a operação da API GetColumnStatisticsForTable.

> **Note**
>
> Se você estiver usando as permissões do Lake Formation para controlar o acesso à tabela, o perfil assumido pela tarefa de estatísticas da coluna exigirá acesso total à tabela para gerar estatísticas.

Tópicos

Pré-requisitos para gerar estatísticas de colunas

Para gerar ou atualizar as estatísticas de colunas, a tarefa de geração de estatísticas assume um perfil do AWS Identity and Access Management (IAM) em seu nome. Com base nas permissões concedidas ao perfil, a tarefa de geração de estatísticas da coluna pode ler os dados do datastore do Amazon S3.

> **Note**
>
> Para gerar estatísticas para tabelas gerenciadas pelo Lake Formation, o perfil do IAM usado para gerar estatísticas exige acesso total à tabela.

Para usar o controle de acesso baseado em perfis, é necessário criar um perfil do IAM com as permissões listadas na política a seguir e adicionar esse perfil à tarefa de geração de estatísticas da coluna.

Para criar um perfil do IAM para gerar estatísticas de coluna

1. Para criar um perfil do IAM, consulte Criar um perfil do IAM para o AWS Glue.
2. Para atualizar um perfil existente, no console do IAM, acesse o perfil do IAM que está sendo usado pelo processo de geração de estatísticas de colunas.
3. Na guia Adicionar permissões, escolha Anexar políticas. Na janela recém-aberta do navegador, escolha política gerenciada pela AWS `AWSGlueServiceRole`.
4. Você também precisa incluir permissão para ler dados da localização de dados do Amazon S3.

   Na seção Adicionar permissões, escolha Criar política. Na janela recém-aberta do navegador, crie uma nova política para usar com seu perfil.
5. Na página Criar política, escolha a guia JSON. Copie o seguinte código `JSON` no editor.

   > **Note**
   >
   > Nas políticas a seguir, substitua o ID da conta por uma Conta da AWS válida e substitua `region` pela região da tabela e `bucket-name` pelo nome do bucket do 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) Se você estiver usando as permissões do Lake Formation para fornecer acesso aos seus dados, o perfil do IAM exigirá permissões de `lakeformation:GetDataAccess`.

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Sid": "LakeFormationDataAccess",
            "Effect": "Allow",
            "Action": "lakeformation:GetDataAccess",
            "Resource": [
                "*"
            ]
        }
    ]
}
```

   Se a localização dos dados do Amazon S3 estiver registrada no Lake Formation e o perfil do IAM assumido pela tarefa de geração de estatísticas de colunas não tiver permissões de grupo `IAM_ALLOWED_PRINCIPALS` concedidas na tabela, o perfil exigirá as permissões `ALTER` e `DESCRIBE` do Lake Formation na tabela. O perfil usado para registrar o bucket do Amazon S3 requer as permissões `INSERT` e `DELETE` do Lake Formation na tabela.

   Se a localização dos dados do Amazon S3 não estiver registrada no Lake Formation e o perfil do IAM não tiver permissões de grupo `IAM_ALLOWED_PRINCIPALS` concedidas na tabela, o perfil exigirá as permissões `ALTER`, `DESCRIBE`, `INSERT` e `DELETE` do Lake Formation na tabela.

7. (Opcional) A tarefa de geração de estatísticas de colunas que grava Amazon CloudWatch Logs criptografados requer as permissões a seguir na política de chave.

```
{
    "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. O perfil que você usa para executar as estatísticas de colunas deve ter a permissão `iam:PassRole`.

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

```
            "iam:PassRole"
        ],
        "Resource": [
            "arn:aws:iam::111122223333:role/<columnstats-role-name>"
        ]
    }]
}
```

9. Ao criar um perfil do IAM para gerar estatísticas de coluna, esse perfil também deve ter a política de confiança a seguir que permite que o serviço assuma o perfil.

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

Gerar estatísticas de colunas

Siga estas etapas para gerenciar a geração de estatísticas no Catálogo de Dados usando o console do AWS Glue ou a AWS CLI.

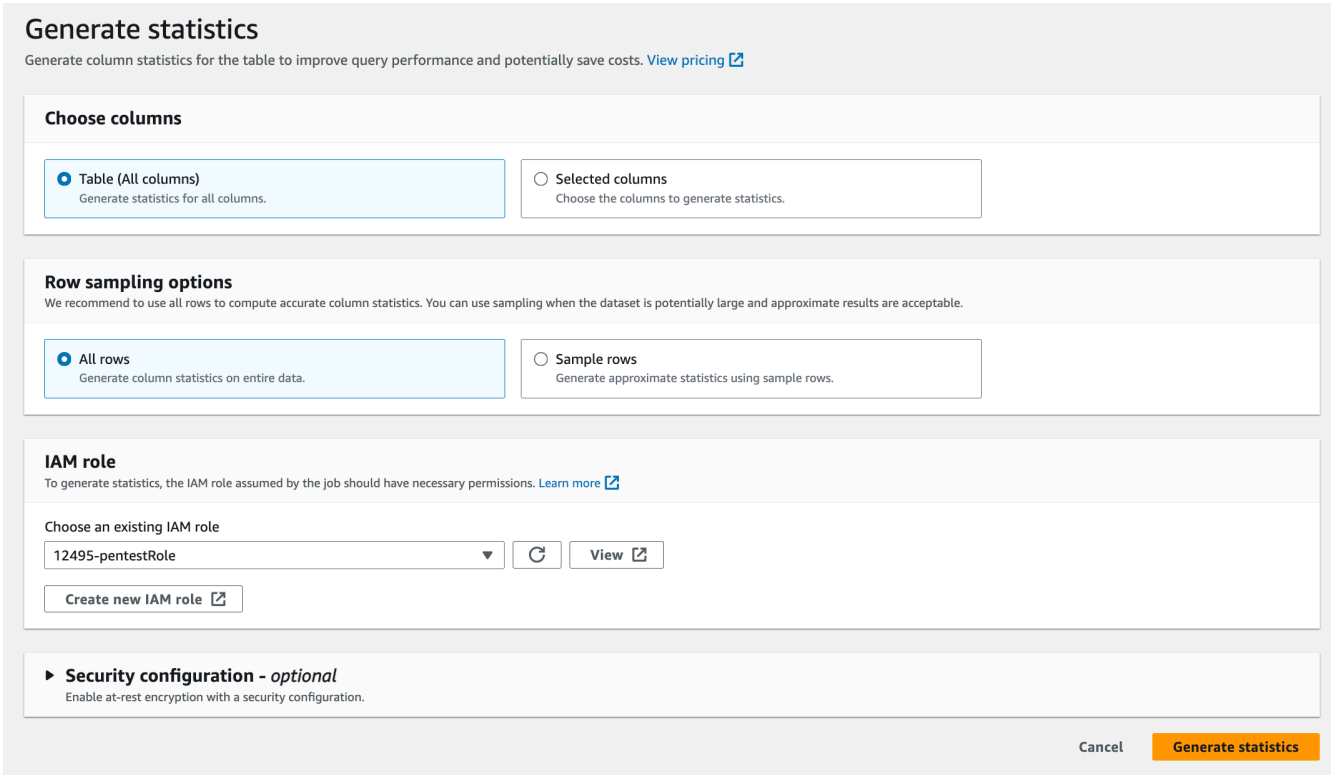
Console

Para gerar estatísticas de colunas usando o console

1. Faça login no console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Escolha uma tabela do Catálogo de Dados.
3. Escolha uma tabela na lista.
4. Escolha Gerar estatísticas no menu Ações.

Você também pode escolher o botão Gerar estatísticas na guia Estatísticas da coluna na seção inferior da página Tabelas.

5. Na página Gerar estatísticas, especifique as seguintes opções:



- Tabela (todas as colunas): escolha essa opção para gerar estatísticas para todas as colunas na tabela.
- Colunas selecionadas: escolha essa opção para gerar estatísticas para colunas específicas. É possível selecionar as colunas na lista suspensa.
- Todas as linhas: escolha todas as linhas da tabela para gerar estatísticas precisas.
- Linhas de exemplo: escolha somente uma porcentagem específica de linhas da tabela para gerar estatísticas. O padrão é todas as linhas. Use as setas para cima e para baixo para aumentar ou diminuir o valor percentual.

> **Note**
>
> Recomendamos incluir todas as linhas na tabela para calcular estatísticas precisas. Use as linhas de exemplo para gerar estatísticas de coluna somente quando valores aproximados forem aceitáveis.

6. (Opcional) Em seguida, escolha uma configuração de segurança para ativar a criptografia em repouso para logs.
7. Escolha Gerar estatísticas para executar o processo.

AWS CLI

No exemplo a seguir, substitua os valores de `DatabaseName`, `TableName` e `ColumnNameList` pelos nomes reais do banco de dados, da tabela e da coluna. Substitua o ID da conta por uma Conta da AWS válida e o nome do perfil pelo nome do perfil do IAM que você está usando para gerar estatísticas.

```
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
}
```

Também é possível pode gerar estatísticas de coluna chamando a operação StartColumnStatisticsTaskRun.

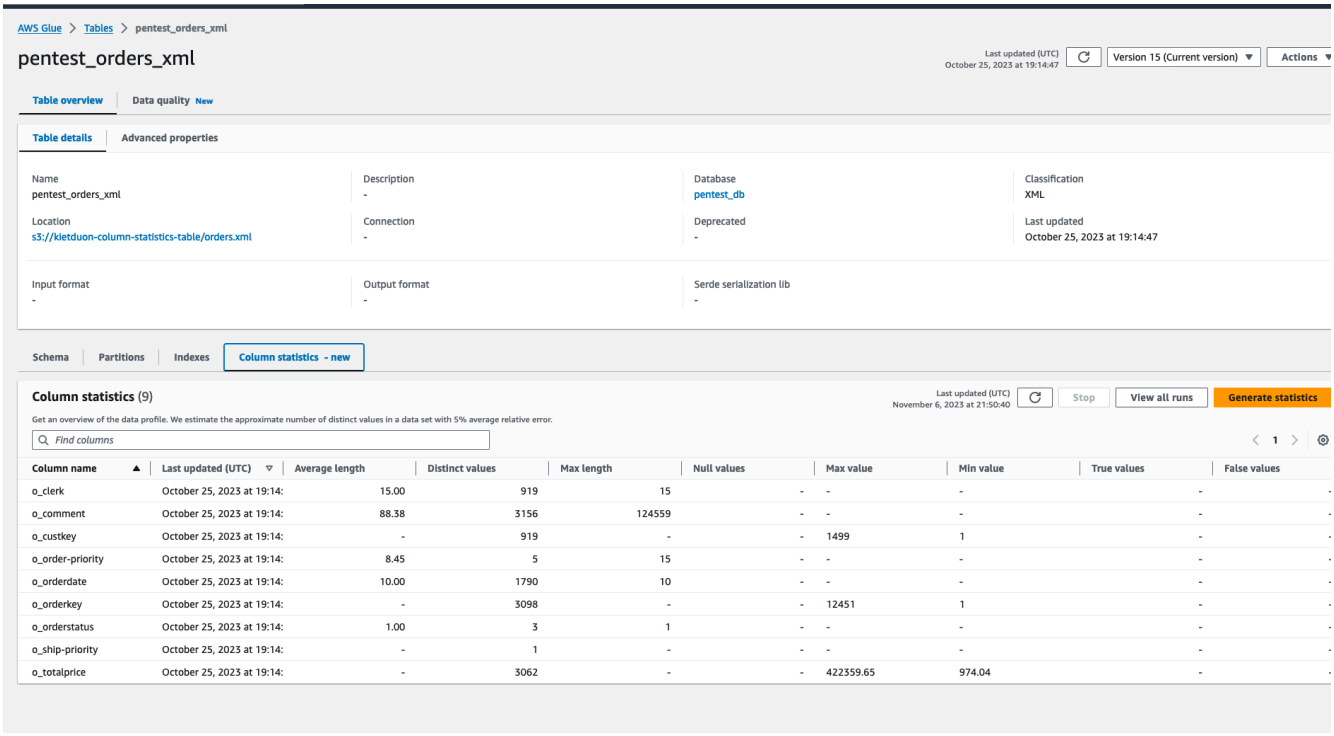
Visualizar estatísticas de colunas

Depois de gerar as estatísticas com êxito, o Data Catalog armazena essas informações para que os otimizadores baseados em custos no Amazon Athena e no Amazon Redshift façam as melhores escolhas ao executar consultas. As estatísticas variam de acordo com o tipo de coluna.

AWS Management Console

Para visualizar estatísticas de colunas para uma tabela

- Após a execução da tarefa de estatísticas da coluna, a guia Estatísticas da coluna na página Detalhes da tabela mostra as estatísticas da tabela.



As seguintes estatísticas estão disponíveis:

- Nome da coluna: o nome da coluna usada para gerar as estatísticas
- Última atualização: a data e a hora em que as estatísticas foram geradas
- Comprimento médio: o comprimento médio dos valores na coluna
- Valores distintos: o número total de valores distintos na coluna. Estimamos o número de valores distintos em uma coluna com erro relativo de 5%.
- Valor máximo: o maior valor na coluna.
- Valor mínimo: o menor valor na coluna.
- Comprimento máximo: o comprimento do valor mais alto na coluna.
- Valores nulos: o número de valores nulos na coluna.
- Valores verdadeiros: o número de valores verdadeiros na coluna.
- Valores falsos: o número de valores falsos na coluna.

AWS CLI

O exemplo a seguir mostra como recuperar estatísticas de colunas usando a AWS CLI.

```
aws glue get-column-statistics-for-table \
```

```
    --database-name <test_db> \
    --table-name <test_tble> \
    --column-names <col1>
```

Você também pode visualizar as estatísticas da coluna usando a operação da API GetColumnStatisticsForTable.

Atualizar estatísticas de colunas

Manter as estatísticas atualizadas melhora a performance das consultas, permitindo que o planejador de consultas escolha os planos ideais. É necessário executar explicitamente a tarefa Gerar estatísticas no console do AWS Glue para atualizar as estatísticas da coluna. O Catálogo de Dados não atualiza as estatísticas automaticamente.

Se você não estiver usando o recurso de geração de estatísticas do AWS Glue no console, poderá atualizar manualmente as estatísticas da coluna usando a operação da API UpdateColumnStatisticsForTable ou a AWS CLI. O exemplo a seguir mostra como excluir estatísticas de colunas usando a 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
                }
            }
        }
    ]
}
```

Excluir estatísticas de colunas

Você pode excluir estatísticas de colunas usando a operação da API DeleteColumnStatisticsForTable ou a AWS CLI. Os exemplos a seguir mostram como excluir estatísticas de colunas usando a AWS Command Line Interface (AWS CLI).

```
aws glue delete-column-statistics-for-table \
    --database-name test_db \
    --table-name test_table \
    --column-name col1
```

Visualizar as execuções de tarefas de estatísticas de colunas

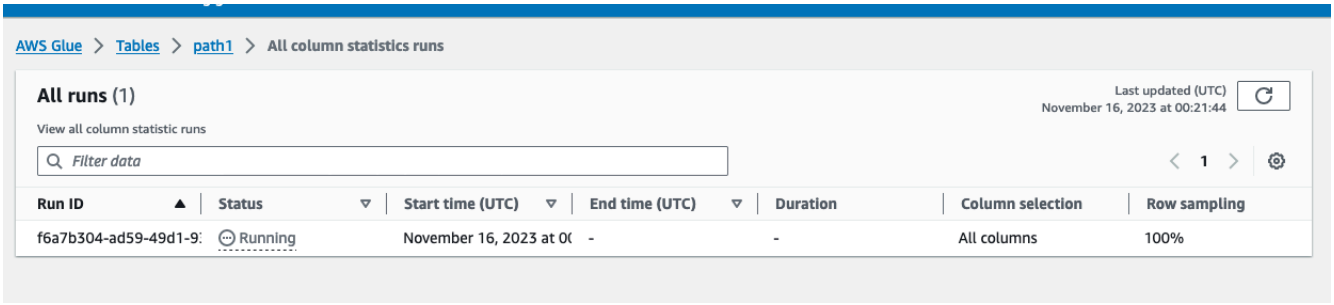
Depois de executar uma tarefa de estatísticas de coluna, é possível explorar os detalhes da execução da tarefa para uma tabela usando o console do AWS Glue, a AWS CLI ou a operação GetColumnStatisticsTaskRuns.

Console

Para visualizar detalhes da execução da tarefa de estatísticas da coluna

1. No console do AWS Glue, escolha Tabelas em Catálogo de Dados.
2. Selecione uma tabela com estatísticas de colunas.
3. Na página Detalhes da tabela, escolha Estatísticas da coluna.
4. Escolha Visualizar execuções.

   É possível ver informações sobre todas as execuções associadas à tabela especificada.



AWS CLI

No exemplo a seguir, substitua os valores de `DatabaseName` e `TableName` pelos nomes reais do banco de dados e da tabela.

```
aws glue get-column-statistics-task-runs --input-cli-json file://input.json
{
    "DatabaseName": "<test-db>",
    "TableName": "<test-table>"
}
```

Interromper a execução da tarefa de estatísticas de coluna

Você pode interromper a execução de uma tarefa de estatísticas de coluna para uma tabela usando o console do AWS Glue, a AWS CLI ou a operação StopColumnStatisticsTaskRun.

Console

Para interromper a execução de uma tarefa de estatísticas de coluna

1. No console do AWS Glue, escolha Tabelas em Catálogo de Dados.
2. Selecione a tabela com a execução da tarefa de estatísticas da coluna em andamento.
3. Na página Detalhes da tabela, escolha Estatísticas da coluna.
4. Escolha Parar.

   Se você interromper a tarefa antes que a execução seja concluída, as estatísticas da coluna não serão geradas para a tabela.

AWS CLI

No exemplo a seguir, substitua os valores de `DatabaseName` e `TableName` pelos nomes reais do banco de dados e da tabela.

```
aws glue stop-column-statistics-task-run --input-cli-json file://input.json
{
    "DatabaseName": "<test-db>",
    "TableName": "<test-table>"
}
```

Considerações e limitações

As considerações e limitações a seguir se aplicam à geração de estatísticas de colunas.

Considerações

- Usar amostragem para gerar estatísticas reduz o tempo de execução, mas pode gerar estatísticas imprecisas.
- Cada execução de estatísticas de coluna exige o processamento de todo o conjunto de dados.
- O Catálogo de Dados não armazena versões diferentes das estatísticas.
- Só é possível executar uma tarefa de geração de estatísticas de cada vez por tabela.
- Se uma tabela for criptografada usando a chave AWS KMS do cliente registrada no Catálogo de Dados, o AWS Glue usa a mesma chave para criptografar as estatísticas.

A tarefa de estatísticas de colunas oferece suporte à geração de estatísticas:

- Quando o perfil do IAM tem permissões completas na tabela (IAM ou Lake Formation).
- Quando o perfil do IAM tem permissões na tabela usando o modo de acesso híbrido do Lake Formation.

A tarefa de estatísticas de colunas não oferece suporte à geração de estatísticas para:

- Tabelas com controle de acesso baseado em células do Lake Formation.
- Data lakes transacionais: Delta Lake do Linux foundation, Apache Iceberg, Apache Hudi.
- Tabelas em bancos de dados federados: metastore do Hive, compartilhamentos de dados do Amazon Redshift
- Colunas aninhadas, matrizes e tipos de dados struct.
- Tabela que é compartilhada com você de outra conta.

Como criptografar seu Data Catalog

Você pode proteger seus metadados armazenados no AWS Glue Data Catalog em repouso usando chaves de criptografia gerenciadas pelo AWS Key Management Service (AWS KMS). É

possível habilitar a criptografia do Catálogo de Dados para o novo Catálogo de Dados usando as Configurações do Catálogo de Dados. Você pode habilitar ou desabilitar a criptografia para o Catálogo de Dados existente conforme necessário. Quando habilitada, o AWS Glue criptografa todos os novos metadados gravados no catálogo, enquanto os metadados existentes permanecem sem criptografia.

Para obter informações detalhadas sobre como criptografar seu Catálogo de Dados, consulte Como criptografar seu Data Catalog.

Proteger seu catálogo de dados usando o Lake Formation

O AWS Lake Formation é um serviço que facilita configurar um data lake seguro na AWS. Ele fornece um local central para criar e gerenciar com segurança seus data lakes com a definição de permissões de controle de acesso refinadas. O Lake Formation usa o Catálogo de Dados para armazenar e recuperar metadados sobre seu data lake, como definições de tabelas, informações de esquema e configurações de controle de acesso a dados.

Você pode registrar a localização da tabela de metadados ou do banco de dados no Amazon S3 com o Lake Formation e usá-la para definir permissões em nível de metadados nos recursos do Catálogo de Dados. Também é possível usar o Lake Formation para gerenciar permissões de acesso ao armazenamento nos dados subjacentes armazenados no Amazon S3 em nome de mecanismos analíticos integrados.

Para obter mais informações, consulte O que é o AWS Lake Formation?.

Acessar o Catálogo de Dados

É possível usar o AWS Glue Data Catalog para descobrir e entender seus dados. O Catálogo de Dados fornece uma maneira consistente de manter definições de esquema, tipos de dados, localizações e outros metadados. É possível acessar o Catálogo de Dados usando qualquer um dos seguintes métodos:

- Console do AWS Glue: é possível acessar e gerenciar o Catálogo de Dados via console do AWS Glue, uma interface do usuário baseada na Web. O console permite que você navegue por e pesquise bancos de dados, tabelas e seus metadados associados, além de criar, atualizar e excluir definições de metadados.
- Crawler do AWS Glue: os crawlers são programas que examinam automaticamente suas fontes de dados e preenchem o Catálogo de Dados com metadados. Você pode criar e executar crawlers para descobrir e catalogar dados de várias fontes, como Amazon S3, Amazon RDS, Amazon

DynamoDB, Amazon CloudWatch e bancos de dados relacionais compatíveis com JDBC, como MySQL e PostgreSQL, além de várias fontes não relacionadas à AWS, como Snowflake e Google BigQuery.

- APIs do AWS Glue: é possível acessar o catálogo de dados programaticamente usando as APIs do AWS Glue. Essas APIs permitem que você interaja com o Catálogo de Dados de forma programática, possibilitando a automação e a integração com outros serviços e aplicações.
- AWS Command Line Interface (AWS CLI): é possível usar a AWS CLI para acessar e gerenciar o Catálogo de Dados a partir da linha de comando. A CLI fornece comandos para criar, atualizar e excluir definições de metadados, bem como consultar e recuperar informações de metadados.
- Integração com outros serviços da AWS: o Catálogo de Dados se integra a vários outros serviços da AWS, permitindo que você acesse e utilize os metadados armazenados no catálogo. Por exemplo, você pode usar o Amazon Athena para consultar fontes de dados usando os metadados no catálogo de dados e usar o AWS Lake Formation para gerenciar o acesso aos dados e a governança dos recursos do Catálogo de Dados.

Práticas recomendadas do Catálogo de Dados do AWS Glue

Esta seção aborda práticas recomendadas para gerenciar e utilizar com eficácia o AWS Glue Data Catalog. Ele enfatiza práticas como uso eficiente de crawlers, organização de metadados, segurança, otimização de performance, automação, governança de dados e integração com outros serviços da AWS.

- Use rastreadores de forma eficaz: execute crawlers regularmente para manter o Catálogo de Dados atualizado com as alterações em suas fontes de dados. Use crawls incrementais para alterar frequentemente as fontes de dados para melhorar a performance. Configure crawlers para adicionar automaticamente novas partições ou atualizar esquemas quando alterações forem detectadas.
- Organize e nomeie tabelas de metadados: estabeleça uma convenção de nomenclatura consistente para bancos de dados e tabelas no Catálogo de Dados. Agrupe fontes de dados relacionadas em bancos de dados ou pastas lógicas para alcançar uma melhor organização. Use nomes descritivos que transmitam a finalidade e o conteúdo de cada tabela.
- Gerencie esquemas de forma eficaz: utilize os recursos de inferência de esquemas dos crawlers do AWS Glue. Revise e atualize alterações no do esquema antes de aplicá-las para evitar a interrupção de aplicações mais à frente. Use os recursos de evolução de esquema para lidar suavemente com alterações no esquema.

- Proteja o Catálogo de Dados: habilite a criptografia de dados em repouso e em trânsito para o Catálogo de Dados. Implemente políticas de controle de acesso refinadas para restringir o acesso a dados confidenciais. Audite e analise regularmente as permissões e os logs de atividades do Catálogo de Dados.
- Integre com outros serviços da AWS Use o Catálogo de Dados como uma camada centralizada de metadados para serviços como Amazon Athena, Redshift Spectrum e AWS Lake Formation. Aproveite os trabalhos do AWS Glue ETL para transformar e carregar dados em vários armazenamentos de dados enquanto mantém os metadados no Catálogo de Dados.
- Monitore e otimize a performance O Catálogo de dados monitora a performance de crawlers e trabalhos de ETL usando métricas do Amazon CloudWatch. Particione grandes conjuntos de dados no Catálogo de Dados para melhorar a performance das consultas. Implemente otimizações de performance para metadados acessados com frequência.
- Mantenha-se em dia com a documentação e as práticas recomendadas do AWS Glue O Catálogo de dados verifica regularmente a documentação do AWS Glue e os recursos do AWS Glue em busca de atualizações, práticas recomendadas e recomendações mais recentes. Participe de webinars, workshops e outros eventos do AWS Glue para aprender com especialistas e se manter em dia com novos recursos e capacidades.

Registro de esquemas do AWS Glue

> **Note**
>
> O registro de esquemas do AWS Glue não é compatível com o console do AWS Glue nas seguintes regiões: Ásia-Pacífico (Jacarta) e Oriente Médio (EAU).

O registro de esquemas do AWS Glue é um novo recurso que permite detectar, controlar e evoluir centralmente esquemas de fluxo de dados. O esquema define a estrutura e o formato de um registro de dados. Com o registro de esquemas do AWS Glue, você pode gerenciar e aplicar esquemas em suas aplicações de fluxo de dados usando integrações convenientes com o Apache Kafka, Amazon Managed Streaming for Apache Kafka, Amazon Kinesis Data Streams, Amazon Managed Service for Apache Flink e AWS Lambda.

O registro de esquemas do AWS Glue suporta o formato de dados AVRO (v1.10.2), o formato de dados JSON com formato de esquema JSON para o esquema (especificações Draft-04, Draft-06 e Draft-07) com validação de esquema JSON usando a biblioteca Everit,Protocol Buffers (Protobuf)

versões proto2 e proto3 sem suporte para `extensions` ou `groups` e suporte à linguagem Java, com outros formatos de dados e linguagens por vir. Os recursos suportados incluem compatibilidade, fornecimento de esquema via metadados, registro automático de esquemas, compatibilidade com o IAM e compactação ZLIB opcional para reduzir o armazenamento e a transferência de dados. O registro de esquemas do AWS Glue é uma solução sem servidor e de uso gratuito.

O uso de um esquema como um contrato de formato de dados entre produtores e consumidores leva a uma melhor governança e maior qualidade dos dados, e ainda permite que os consumidores de dados sejam resilientes a alterações upstream compatíveis.

O registro de esquemas permite que sistemas diferentes compartilhem um esquema para serialização e desserialização. Por exemplo, suponha que você tenha um produtor e consumidor de dados. O produtor conhece o esquema quando publica os dados. O registro de esquemas fornece um serializador e desserializador para determinados sistemas, como Amazon MSK ou Apache Kafka.

Para ter mais informações, consulte Como funciona o registro de esquemas.

Tópicos

Esquemas

O esquema define a estrutura e o formato de um registro de dados. Um esquema é uma especificação versionada para publicação, consumo ou datastore confiáveis.

Neste esquema de exemplo para Avro, o formato e a estrutura são definidos pelo layout e nomes de campo, e o formato dos nomes de campo é definido pelos tipos de dados (por exemplo,`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"
                    }
                ]
            }
        }
    ]
}
```

Neste exemplo de esquema JSON Draft-07 para JSON, o formato é definido pela organização do 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
  }
 }
}
```

Neste exemplo para Protobuf, o formato é definido pela versão 2 da linguagem Protocol Buffers (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;
}
```

Registros

Um registro é um contêiner lógico de esquemas. Os registros permitem que você organize seus esquemas, bem como gerencie o controle de acesso para suas aplicações. Um registro tem um nome do recurso da Amazon (ARN) para permitir que você organize e defina diferentes permissões de acesso para operações de esquema dentro do registro.

Você pode usar o registro padrão ou criar quantos novos registros forem necessários.

Hierarquia do registro de esquemas do AWS Glue

- RegistryName: [string]
  - RegistryArn: [ARN da AWS]
  - CreatedTime: [carimbo de data e hora]
  - UpdatedTime: [carimbo de data e hora]
- SchemaName: [string]
  - SchemaArn: [ARN da AWS]
  - DataFormat: [Avro, Json ou Protobuf]
  - Compatibility: [ex.: BACKWARD, BACKWARD_ALL, FORWARD, FORWARD_ALL, FULL, FULL_ALL, NONE, DISABLED]
  - Status: [ex.: PENDING, AVAILABLE, DELETING]
  - SchemaCheckpoint: [inteiro]
  - CreatedTime: [carimbo de data e hora]
  - UpdatedTime: [carimbo de data e hora]
- SchemaVersion: [string]
  - SchemaVersionNumber: [inteiro]
  - Status: [ex.: PENDING, AVAILABLE, DELETING, FAILURE]
  - SchemaDefinition: [string, Valor: JSON]

- CreatedTime: [carimbo de data e hora]

- SchemaVersionMetadata: [lista]
  - MetadataKey: [string]
  - MetadataInfo
  - MetadaValue: [string]
  - CreatedTime: [carimbo de data e hora]

Versionamento e compatibilidade de esquema

Cada esquema pode ter várias versões. O versionamento é regido por uma regra de compatibilidade aplicada em um esquema. As solicitações para registrar novas versões do esquema são verificadas quanto a essa regra pelo registro de esquemas antes de serem bem-sucedidas.

Uma versão de esquema que está marcada como um ponto de verificação é usada para determinar a compatibilidade em registrar novas versões de um esquema. Quando um esquema é criado pela primeira vez, o ponto de verificação padrão é a primeira versão. À medida que o esquema evolui com mais versões, você pode usar a CLI/SDK para alterar o ponto de verificação para uma versão de um esquema usando a API `UpdateSchema` que adere a um conjunto de restrições. No console, editar a definição do esquema ou o modo de compatibilidade alterará o ponto de verificação para a versão mais recente por padrão.

Os modos de compatibilidade permitem controlar como os esquemas podem ou não evoluir ao longo do tempo. Esses modos formam o contrato entre aplicações que produzem e consomem dados. Quando uma nova versão de um esquema é enviada para o registro, a regra de compatibilidade aplicada ao nome do esquema é usada para determinar se a nova versão pode ser aceita. Existem oito modos de compatibilidade: NONE (Nenhum), DISABLED (Desabilitado), BACKWARD (Anterior), BACKWARD_ALL (Todas as anteriores), FORWARD (Próxima), FORWARD_ALL (Todas as próximas), FULL (Completo), FULL_ALL (Completo total).

No formato de dados Avro, os campos podem ser opcionais ou obrigatórios. Um campo opcional é aquele em que o `Type` inclui nulo. Campos obrigatórios não possuem nulo como o `Type`.

No formato de dados Protobuf, os campos podem ser opcionais (inclusive repetidos) ou obrigatórios na sintaxe proto2, enquanto todos os campos são opcionais (inclusive repetidos) na sintaxe proto3. Todas as regras de compatibilidade são determinadas com base no entendimento das

especificações de Protocol Buffers, bem como na orientação da Documentação do Google Protocol Buffers.

- NONE (Nenhum): nenhum modo de compatibilidade se aplica. Você pode usar essa opção em cenários de desenvolvimento ou se não souber os modos de compatibilidade que deseja aplicar aos esquemas. Qualquer nova versão adicionada será aceita sem ser submetida a uma verificação de compatibilidade.
- DISABLED (Desabilitado): essa opção de compatibilidade impede o versionamento de um esquema específico. Nenhuma nova versão pode ser adicionada.
- BACKWARD (Anterior): essa opção de compatibilidade é recomendada, pois permite que os consumidores de dados leiam a versão atual e anterior do esquema. Você pode usar essa opção para verificar a compatibilidade com a versão anterior do esquema ao excluir campos ou adicionar campos opcionais. Um caso de uso típico para BACKWARD (Anterior) é quando sua aplicação é criada para o esquema mais recente.

  AVRO

  Por exemplo, suponha que você tenha um esquema definido pelo nome (obrigatório), sobrenome (obrigatório), e-mail (obrigatório) e número de telefone (opcional).

  Se a próxima versão do esquema removesse o campo de e-mail obrigatório, isso seria registrado com êxito. A compatibilidade BACKWARD (Anterior) requer que os consumidores sejam capazes de ler a versão atual e anterior do esquema. Seus consumidores poderão ler o novo esquema, pois o campo de e-mail extra de mensagens antigas será ignorado.

  Se você tivesse uma nova versão de esquema proposta que adicionasse um campo obrigatório, por exemplo, CEP, isso não seria registrado com êxito com a compatibilidade BACKWARD (Anterior). Seus consumidores na nova versão não conseguiriam ler mensagens antigas antes da alteração do esquema, pois elas não teriam o campo CEP necessário. No entanto, se o campo CEP fosse definido como opcional no novo esquema, a versão proposta seria registrada com sucesso, pois os consumidores poderiam ler o esquema antigo sem o campo de CEP opcional.

  JSON

  Por exemplo, suponha que você tenha uma versão do esquema definida pelo nome (opcional), sobrenome (opcional), e-mail (opcional) e número de telefone (opcional).

  Se a próxima versão do esquema adicionasse a propriedade de número de telefone opcional, isso seria registrado com êxito, desde que a versão do esquema original não permitisse nenhuma

propriedade adicional definindo o campo `additionalProperties` como false. A compatibilidade BACKWARD (Anterior) requer que os consumidores sejam capazes de ler a versão atual e anterior do esquema. Seus consumidores poderão ler dados produzidos com o esquema original onde a propriedade de número de telefone não existe.

Se você tiver uma nova versão de esquema proposta que adiciona a propriedade de número de telefone opcional, isso não é registrado com êxito com a compatibilidade BACKWARD (Anterior) quando a versão original do esquema define o campo `additionalProperties` como true, ou seja, permitindo qualquer propriedade adicional. Seus consumidores na nova versão não conseguiriam ler mensagens antigas antes da alteração do esquema, pois não podem ler dados com a propriedade número de telefone em um tipo diferente, por exemplo string em vez de número.

PROTOBUF

Por exemplo, suponha que você tenha uma versão do esquema definida por uma Mensagem `Person` com campos `first name` (obrigatório), `last name` (obrigatório), `email` (obrigatório) e `phone number` (opcional) sob sintaxe proto2.

Semelhante a cenários AVRO, se a próxima versão do esquema removesse o campo de `email` obrigatório, isso seria registrado com êxito. A compatibilidade BACKWARD (Anterior) requer que os consumidores sejam capazes de ler a versão atual e anterior do esquema. Seus consumidores poderão ler o novo esquema, pois o campo de `email` extra de mensagens antigas será ignorado.

Se você tivesse uma nova versão de esquema proposta que adicionasse um campo obrigatório, por exemplo, `zip code`, isso não seria registrado com êxito com a compatibilidade BACKWARD (Anterior). Seus consumidores na nova versão não conseguiriam ler mensagens antigas antes da alteração do esquema, pois elas não teriam o campo `zip code` necessário. No entanto, se o campo `zip code` fosse definido como opcional no novo esquema, a versão proposta seria registrada com sucesso, pois os consumidores poderiam ler o esquema antigo sem o campo de `zip code` opcional.

No caso de um caso de uso gRPC, adicionar novo serviço RPC ou método RPC é uma alteração compatível com versões anteriores. Por exemplo, suponha que você tenha uma versão do esquema definida por um serviço RPC `MyService` com dois métodos RPC `Foo` e `Bar`.

Se a próxima versão do esquema adicionar um novo método RPC chamado `Baz`, isso seria registrado com sucesso. Seus consumidores poderão ler dados produzidos com o esquema

original de acordo com a compatibilidade BACKWARD pois o método RPC recém-adicionado `Baz` é opcional.

Se você tivesse uma nova versão de esquema proposta que removesse o método PRC existente `Foo`, isso não seria registrado com êxito com a compatibilidade BACKWARD. Seus consumidores na nova versão não conseguiriam ler mensagens antigas antes da alteração do esquema, pois não conseguem entender e ler dados com o método RPC `Foo` inexistente em uma aplicação gRPC.

- BACKWARD_ALL (Todas as anteriores): essa opção de compatibilidade é recomendada, pois permite que os consumidores de dados leiam a versão atual e todas as anteriores do esquema. Você pode usar essa opção para verificar a compatibilidade com todas as versões anteriores do esquema ao excluir campos ou adicionar campos opcionais.
- FORWARD (Próxima): essa opção de compatibilidade permite que os receptores de dados leiam a versão atual e as versões subsequentes do esquema, mas não necessariamente versões mais recentes. Você pode usar essa opção para verificar a compatibilidade com a última versão do esquema ao adicionar campos ou excluir campos opcionais. Um caso de uso típico para FORWARD (Próxima) é quando sua aplicação é criada para um esquema anterior e deve ser capaz de processar um esquema mais recente.

  AVRO

  Por exemplo, suponha que você tenha uma versão do esquema definida pelo nome (obrigatório), sobrenome (obrigatório) e e-mail (opcional).

  Se você tivesse uma nova versão do esquema que adicionasse um campo obrigatório, por exemplo, número de telefone, isso seria registrado com sucesso. A compatibilidade FORWARD (Próxima) requer que os consumidores sejam capazes de ler dados produzidos com o novo esquema usando ainda a versão anterior.

  Se você tivesse uma nova versão de esquema proposta que excluísse o campo de nome obrigatório, isso não seria registrado com êxito com a compatibilidade FORWARD (Posterior). Seus consumidores na versão anterior não seriam capazes de ler os esquemas propostos, pois eles não teriam o campo de nome obrigatório. No entanto, se o campo de nome fosse originalmente opcional, então o novo esquema proposto seria registrado com sucesso, pois os consumidores poderiam ler dados com base no novo esquema que não tem o campo de nome opcional.

  JSON

Por exemplo, suponha que você tenha uma versão do esquema definida pelo nome (opcional), sobrenome (opcional), e-mail (opcional) e número de telefone (opcional).

Se a próxima versão do esquema removesse a propriedade de número de telefone opcional, isso seria registrado com êxito, desde que a versão do esquema original não permitisse nenhuma propriedade adicional definindo o campo `additionalProperties` como false. A compatibilidade FORWARD (Próxima) requer que os consumidores sejam capazes de ler dados produzidos com o novo esquema usando ainda a versão anterior.

Se você tiver uma versão de esquema proposta que exclui a propriedade número de telefone opcional, isso não é registrado com êxito com a compatibilidade FORWARD (Próxima) quando a nova versão do esquema define o campo `additionalProperties` como true, ou seja, permitindo qualquer propriedade adicional. Seus consumidores na versão anterior não conseguiriam ler mensagens antigas antes da alteração do esquema, pois não poderiam ler dados com a propriedade número de telefone em um tipo diferente, por exemplo string em vez de número.

PROTOBUF

Por exemplo, suponha que você tenha uma versão do esquema definida por uma Mensagem `Person` com campos `first name` (obrigatório), `last name` (obrigatório) e `email` (opcional) sob sintaxe proto2.

Semelhante a cenários AVRO, se você tivesse uma nova versão do esquema que adicionasse um campo obrigatório, por exemplo, `phone number`, isso seria registrado com sucesso. A compatibilidade FORWARD (Próxima) requer que os consumidores sejam capazes de ler dados produzidos com o novo esquema usando ainda a versão anterior.

Se você tivesse uma nova versão de esquema proposta que excluísse o campo de `first name`, isso não seria registrado com êxito com a compatibilidade FORWARD. Seus consumidores na versão anterior não seriam capazes de ler os esquemas propostos, pois eles não teriam o campo de `first name` obrigatório. No entanto, se o campo de `first name` fosse originalmente opcional, então o novo esquema proposto seria registrado com sucesso, pois os consumidores poderiam ler dados com base no novo esquema que não tem o campo de `first name` opcional.

No caso de um caso de uso gRPC, remover um serviço RPC ou método RPC é uma alteração compatível com o encaminhamento. Por exemplo, suponha que você tenha uma versão do esquema definida por um serviço RPC `MyService` com dois métodos RPC `Foo` e `Bar`.

Se a próxima versão do esquema excluir o método RPC existente chamado `Foo`, isso seria registrado com sucesso de acordo com a compatibilidade FORWARD, pois os consumidores podem ler dados produzidos com o novo esquema usando a versão anterior. Se você tivesse uma nova versão de esquema proposta que adicionasse o método RPC `Baz`, isso não seria registrado com êxito com a compatibilidade FORWARD. Seus consumidores na versão anterior não seriam capazes de ler os esquemas propostos, pois eles não teriam o método RPC `Baz` ausente.

- FORWARD_ALL (Todas as próximas): essa opção de compatibilidade permite que os consumidores leiam dados escritos por produtores de qualquer novo esquema registrado. Você pode usar essa opção quando precisar adicionar campos ou excluir campos opcionais e verificar a compatibilidade com todas as versões anteriores do esquema.
- FULL (Completo): essa opção de compatibilidade permite que os consumidores leiam dados gravados por produtores usando a versão anterior ou seguinte do esquema, mas não versões anteriores ou posteriores. Você pode usar essa opção para verificar a compatibilidade com a última versão do esquema ao adicionar ou remover campos opcionais.
- FULL_ALL (Completo total): essa opção de compatibilidade permite que os receptores de dados leiam dados gravados por produtores usando todas as versões de esquema anteriores. Você pode usar essa opção para verificar a compatibilidade com todas as versões anteriores do esquema ao adicionar ou remover campos opcionais.

Bibliotecas Serde de código aberto

A AWS fornece bibliotecas Serde de código aberto como um framework para serialização e desserialização de dados. O design de código aberto dessas bibliotecas permite que aplicações e frameworks comuns de código aberto ofereçam suporte a elas em seus projetos.

Para obter mais detalhes sobre como as bibliotecas Serde funcionam, consulte Como funciona o registro de esquemas.

Cotas do registro do esquemas

As cotas, também conhecidas como limites na AWS, são os valores máximos para recursos, ações e itens na sua conta da AWS. A seguir estão os limites flexíveis para o registro de esquemas no AWS Glue.

Pares de chave-valor de metadados de versão de esquema

Você pode ter até dez pares de chave-valor por SchemaVersion por região da AWS.

Você pode visualizar ou definir os pares de metadados de chave-valor usando o QuerySchemaVersionMetadata ação (Python: query_schema_version_metadata) ou APIs do PutSchemaVersionMetadata ação (Python: put_schema_version_metadata).

A seguir estão os limites rígidos para o registro de esquemas no AWS Glue.

Registros

É possível ter até 100 registros por região da AWS para esta conta.

SchemaVersion

É possível ter até 10.000 versões de esquema por região da AWS para esta conta.

Cada novo esquema cria uma nova versão do esquema, portanto você poderá, teoricamente, ter até 10.000 esquemas por conta por região, se cada esquema tiver apenas uma versão.

Cargas úteis de esquema

Há um limite de tamanho de 170 KB para cargas úteis de esquema.

Como funciona o registro de esquemas

Esta seção descreve como funcionam os processos de serialização e desserialização no registro de esquemas.

1. Registrar um esquema: se o esquema ainda não existir no registro, ele pode ser registrado com um nome de esquema igual ao nome do destino (por exemplo, test_topic, test_stream, prod_firehose) ou o produtor pode fornecer um nome personalizado para ele. Os produtores também podem adicionar pares de chave-valor ao esquema como metadados, como a fonte: msk_kafka_topic_a ou aplicar tags da AWS para esquemas na criação do esquema. Depois que um esquema é registrado, o registro de esquemas retorna o ID da versão do esquema para o serializador. Se o esquema existir, mas o serializador estiver usando uma nova versão que não existe, o registro de esquemas verificará a referência do esquema a uma regra de compatibilidade, para garantir que a nova versão seja compatível antes de registrá-la como uma nova versão.

   Existem dois métodos de registro de um esquema: manual e automático. Você pode registrar um esquema manualmente por meio do console do AWS Glue ou CLI/SDK.

   Quando o registro automático é ativado nas configurações do serializador, o registro automático do esquema é realizado. Se `REGISTRY_NAME` não for fornecido nas configurações do produtor,

o registro automático registrará a nova versão do esquema no registro padrão (default-registry). Consulte Instalar as bibliotecas SerDe para obter informações sobre como especificar a propriedade de registro automático.

2. O serializador valida registros de dados em relação ao esquema: quando a aplicação que produz os dados registra seu esquema, o serializador do registro de esquemas valida que o registro sendo produzido pela aplicação é estruturado com os campos e os tipos de dados correspondentes a um esquema registrado. Se o esquema do registro não corresponder a um esquema registrado, o serializador retornará uma exceção e a aplicação falhará em entregar o registro ao destino.

   Se nenhum esquema existir e o nome do esquema não for fornecido por meio das configurações do produtor, o esquema será criado com o mesmo nome que o nome do tópico (se Apache Kafka ou Amazon MSK) ou nome da transmissão (se Kinesis Data Streams).

   Cada registro tem uma definição de esquema e dados. A definição de esquema é consultada em relação aos esquemas e versões existentes no registro de esquemas.

   Por padrão, os produtores armazenam em cache as definições de esquema e IDs de versão de esquemas dos esquemas registrados. Se a definição da versão do esquema de um registro não corresponder ao que está disponível no cache, o produtor tentará validar o esquema com o registro de esquemas. Se a versão do esquema for válida, seu ID de versão e definição serão armazenados em cache localmente no produtor.

   Você pode ajustar o período de cache padrão (24 horas) dentro das propriedades opcionais do produtor na etapa n.º 3 de Instalar as bibliotecas SerDe.

3. Serializar e entregar registros: se o registro estiver em conformidade com o esquema, o serializador decorará cada registro com o ID da versão do esquema, serializará o registro com base no formato de dados selecionado (AVRO, JSON ou Protobuf. Outros formatos em breve), compactará o registro (configuração opcional do produtor) e o entregará ao destino.

4. Os consumidores desserializam os dados: os consumidores que leem esses dados usam a biblioteca de desserialização do registro de esquemas, que analisa o ID da versão do esquema na carga útil do registro.

5. O desserializador pode solicitar o esquema do registro do esquemas: se esta for a primeira vez que o desserializador viu registros com um ID de versão do esquema específico, o desserializador, usando o ID da versão do esquema, solicitará o esquema do registro do esquemas e o armazenará localmente no consumidor. Se o registro de esquemas não puder

desserializar o registro, o consumidor poderá registrar em log os dados do registro e seguir em frente, ou interromper a aplicação.

6. O desserializador usa o esquema para desserializar o registro: quando o desserializador recupera o ID da versão do esquema do registro do esquemas, ele descompacta o registro (se o registro enviado pelo produtor for compactado) e usa o esquema para o desserializar. Então, a aplicação processa o registro.

> **Note**
>
> Criptografia: seus clientes se comunicam com o registro de esquemas por meio de chamadas de API que criptografam dados em trânsito usando criptografia TLS em HTTPS. Os esquemas armazenados no registro de esquemas são sempre criptografados em repouso usando uma chave do AWS Key Management Service (AWS KMS) gerenciada por um serviço.

> **Note**
>
> Autorização do usuário: o registro de esquemas é compatível com políticas do IAM baseadas em identidade.

Conceitos básicos do registro de esquemas

As seções a seguir fornecem uma visão geral e orientações sobre como configurar e usar o registro de esquemas. Para obter informações sobre conceitos e componentes do registro de esquemas, consulte Registro de esquemas do AWS Glue.

Tópicos

- Instalar as bibliotecas SerDe
- Usar a AWS CLI para APIs do registro de esquemas do AWS Glue
- Criar um registro
- Lidar com um registro específico (JAVA POJO) para JSON
- Criar um esquema
- Atualizar um esquema ou registro

- Excluir um esquema ou registro
- Exemplos do IAM para serializadores
- Exemplos do IAM para desserializadores
- Conectividade privada usando AWS PrivateLink
- Acessar métricas do Amazon CloudWatch
- Exemplo do modelo do AWS CloudFormation para o registro de esquemas

Instalar as bibliotecas SerDe

> **Note**
>
> Pré-requisitos: antes de concluir as etapas a seguir, é necessário ter um cluster do Amazon Managed Streaming for Apache Kafka (Amazon MSK) ou do Apache Kafka em execução. Seus produtores e consumidores precisam estar em execução em Java 8 ou superior.

As bibliotecas SerDe fornecem um framework para serialização e desserialização de dados.

Você instalará o serializador de código aberto para suas aplicações que produzem dados (coletivamente os “serializadores”). O serializador manipula serialização, compactação e interação com o registro de esquemas. O serializador extrai automaticamente o esquema de um registro sendo gravado em um destino compatível com o registro de esquemas, como o Amazon MSK. Da mesma forma, você instalará o desserializador de código aberto em suas aplicações que consomem dados.

Para instalar as bibliotecas em produtores e consumidores:

1. Dentro dos arquivos pom.xml, tanto dos produtores quanto dos consumidores, adicione essa dependência com o código abaixo:

   ```
   <dependency>
       <groupId>software.amazon.glue</groupId>
       <artifactId>schema-registry-serde</artifactId>
       <version>1.1.5</version>
   </dependency>
   ```

   Como alternativa, você pode clonar o repositório do registro de esquemas do AWS Glue do GitHub.

2. Configure seus produtores com estas propriedades obrigatórias:

```
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"
```

   Se não houver esquemas existentes, o registro automático precisará ser ativado (próxima etapa). Se você tiver um esquema que gostaria de aplicar, substitua “my-schema” (meu esquema) pelo nome do seu esquema. Além disso, o “registry-name” (nome do registro) deve ser fornecido se o registro automático do esquema estiver desativado. Se o esquema é criado sob o “default-registry” (registro padrão), o nome do registro pode ser omitido.

3. (Opcional) defina qualquer uma destas propriedades opcionais do produtor. Para obter descrições detalhadas das propriedades, consulte o arquivo ReadMe.

```
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
```

   O registro automático registra a versão do esquema no registro padrão (“default-registry”). Se um `SCHEMA_NAME` não for especificado na etapa anterior, então o nome do tópico será inferido como `SCHEMA_NAME`.

Consulte Versionamento e compatibilidade de esquema, para obter mais informações sobre os modos de compatibilidade.

4. Configure seus consumidores com estas propriedades obrigatórias:

```
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 Região da
 AWS
props.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName()); // Only required for AVRO data format
```

5. (Opcional) defina essas propriedades opcionais do consumidor. Para obter descrições detalhadas das propriedades, consulte o arquivo ReadMe.

```
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
```

Usar a AWS CLI para APIs do registro de esquemas do AWS Glue

Para usar a AWS CLI para APIs do registro de esquemas do AWS Glue, certifique-se de atualizar a AWS CLI para a versão mais recente.

Criar um registro

Você pode usar o registro padrão ou criar quantos novos registros forem necessários usando as APIS do AWS Glue ou o console do AWS Glue.

APIs do AWS Glue

Você pode usar estas etapas para executar essa tarefa usando as APIs do AWS Glue.

Para adicionar um novo registro, use a API do CreateRegistry ação (Python: create_registry). Especifique `RegistryName` como o nome do registro a ser criado, com um comprimento máximo de 255, contendo apenas letras, números, hífens, sublinhados, cifrões ou marcas de hash.

Especifique `Description` como uma string com não mais do que 2.048 bytes de comprimento, correspondente ao padrão de string com várias linhas de endereço URI.

Como alternativa, especifique um ou mais `Tags` para seu registro, como uma matriz de mapa de pares de chave-valor.

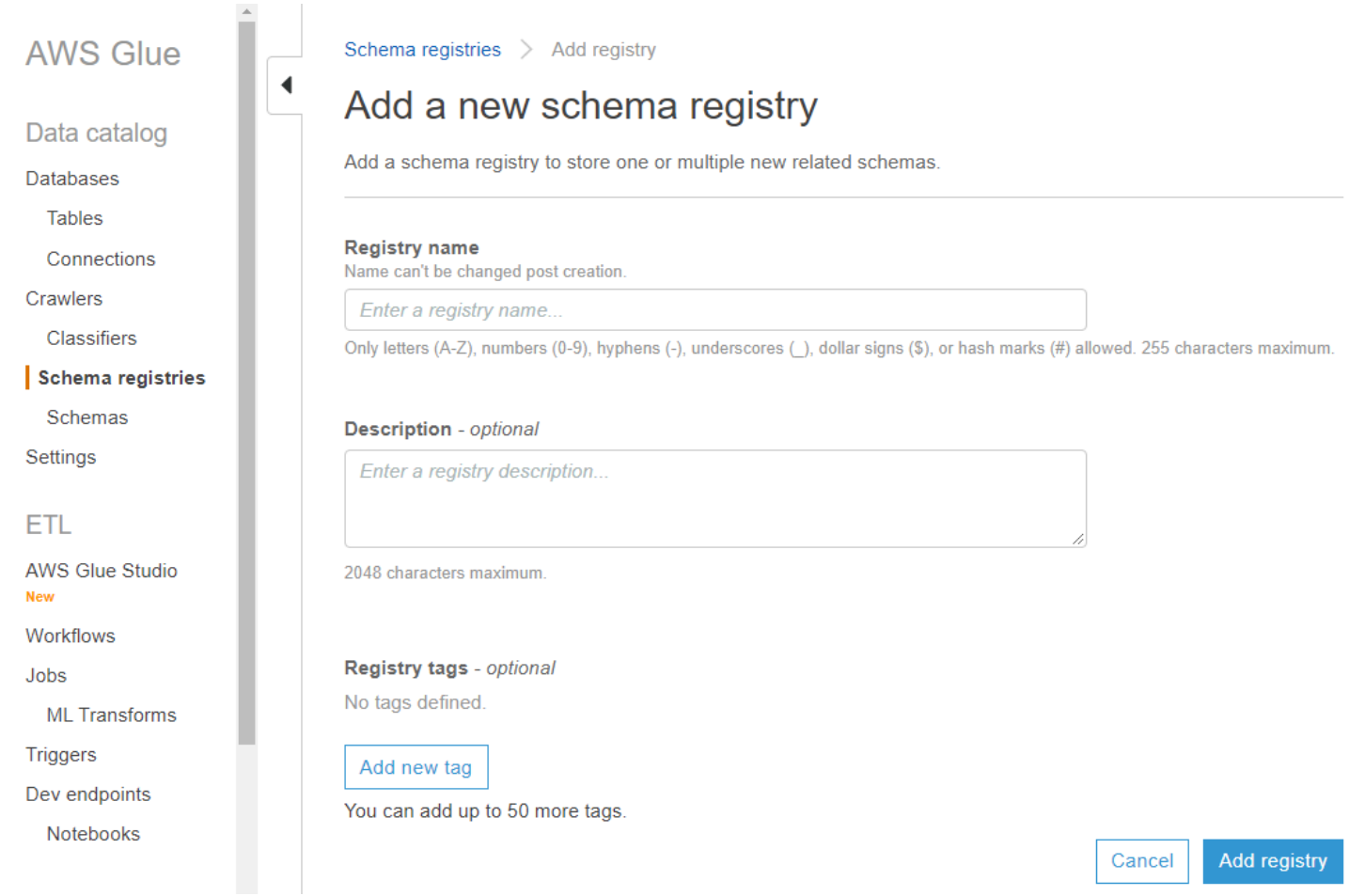
```
aws glue create-registry --registry-name registryName1 --description description
```

Quando seu registro é criado, ele recebe um nome do recurso da Amazon (ARN), que você pode visualizar no `RegistryArn` da resposta da API. Agora que você criou um registro, crie um ou mais esquemas para esse registro.

Console do AWS Glue

Para adicionar um novo registro no console do AWS Glue:

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (Catálogo de dados), escolha Schema registries (Registros de esquemas).
3. Escolha Add registry (Adicionar registro).
4. Digite um Registry name (Nome de registro) para o registro composto por letras, números, hifens ou sublinhados. Esse nome não pode ser alterado.
5. Digite uma Description (Descrição) (opcional) para o registro.
6. Opcionalmente, aplique uma ou mais tags ao registro. Escolha Add new tag (Adicionar nova tag), especifique um Tag key (Chave de tag) e, opcionalmente, um Tag value (Valor da tag).
7. Escolha Add registry (Adicionar registro).



Quando seu registro é criado, ele recebe um nome do recurso da Amazon (ARN), que você pode visualizar escolhendo o registro na lista em Schema registries (Registros de esquemas). Agora que você criou um registro, crie um ou mais esquemas para esse registro.

Lidar com um registro específico (JAVA POJO) para JSON

Você pode usar um plain old Java object (POJO) e transmitir o objeto como um registro. Isso é semelhante à noção de um registro específico no AVRO. O mbknor-jackson-jsonschema pode gerar um esquema JSON para o POJO transmitido. Essa biblioteca também pode injetar informações adicionais no esquema JSON.

A biblioteca do registro de esquemas do AWS Glue usa o campo “className” injetado no esquema para fornecer um nome de classe totalmente classificado. O campo “className” é usado pelo desserializador para desserializar em um objeto dessa classe.

```
 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() {}
}
```

Criar um esquema

Você pode criar um esquema usando as APIs do AWS Glue ou o console do AWS Glue.

APIs do AWS Glue

Você pode usar estas etapas para executar essa tarefa usando as APIs do AWS Glue.

Para adicionar um novo esquema, use a API CreateSchema ação (Python: create_schema).

Especifique uma estrutura `RegistryId` para indicar um registro para o esquema. Ou omita o `RegistryId` para usar o registro padrão.

Especifique um `SchemaName` consistindo em letras, números, hifens ou sublinhados e `DataFormat` como **`AVRO`** ou **`JSON`**. O `DataFormat`, uma vez definido em um esquema, não pode ser alterado.

Especifique um modo de `Compatibility`:

- Backward (Anterior) (recomendado): o consumidor pode ler a versão atual e a anterior.
- Backward all (Todas as anteriores): o consumidor pode ler a versão atual e todas as anteriores.
- Forward (Próxima): o consumidor pode ler a versão atual e a subsequente.
- Forward all (Todas as próximas): o consumidor pode ler a versão atual e todas as subsequentes.
- Full (Completo): combinação de Backward e Forward.
- Full all (Completo total): combinação de Backward all e Forward all.
- None (Nenhum): nenhuma verificação de compatibilidade é realizada.
- Disabled (Desabilitado): impede qualquer versionamento para esse esquema.

Opcionalmente, especifique `Tags` para seu esquema.

Especifique uma `SchemaDefinition` para definir o esquema no formato de dados Avro, JSON ou Protobuf. Consulte os exemplos.

Para o formato de dados 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 o formato de dados 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 o formato de dados Protobuf:

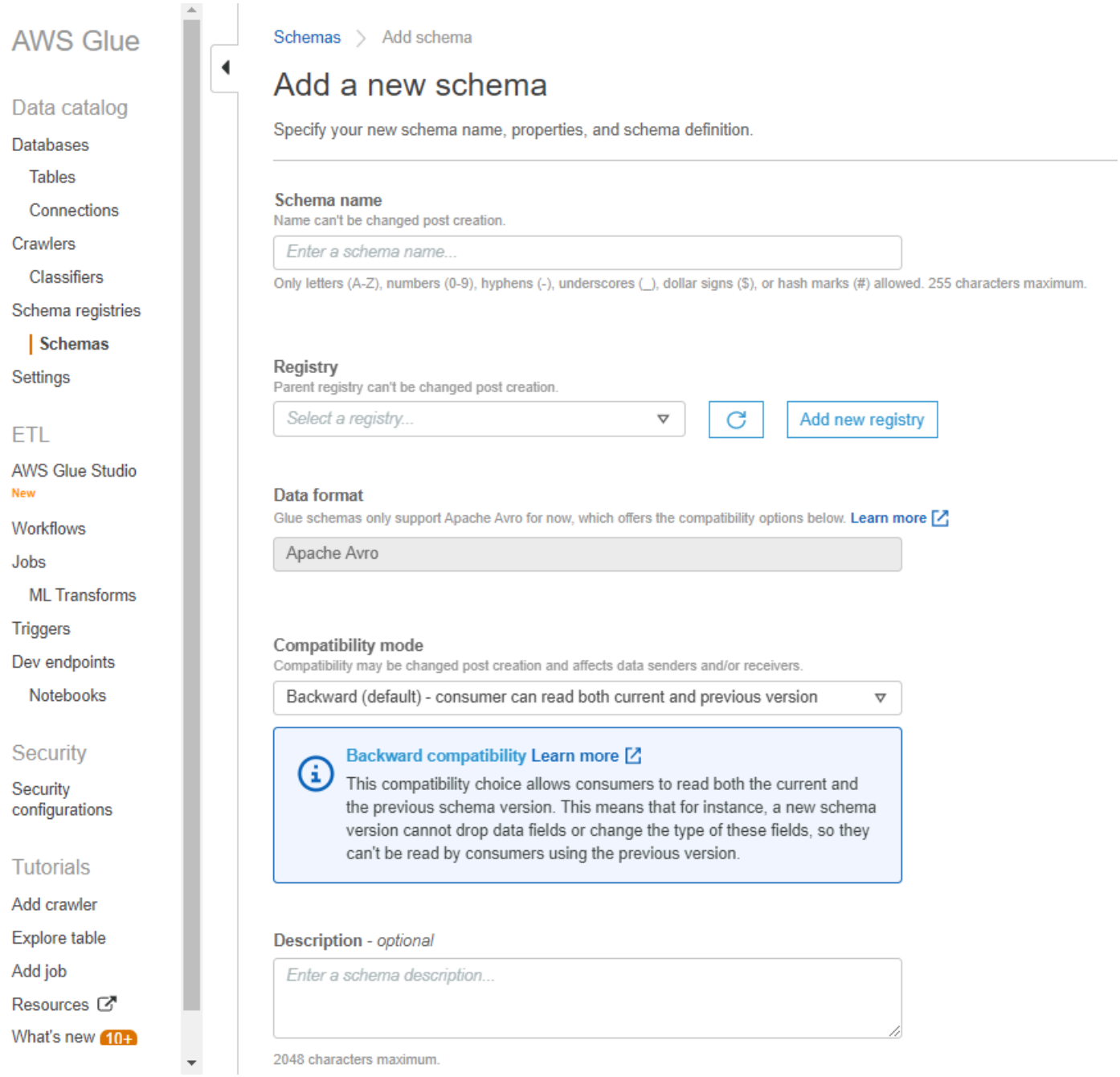
```
aws glue create-schema --registry-id RegistryName="registryName" --schema-name
 testSchemaProtobuf --compatibility NONE --data-format PROTOBUF --schema-definition
 "syntax = \"proto2\";package org.test;message Basic { optional int32 basic = 1;}"
```

```
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;}"
```

Console do AWS Glue

Para adicionar um novo esquema usando o console do AWS Glue:

1. Faça login no Console de Gerenciamento da AWS e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (Catálogo de dados), escolha Schema (Esquema).
3. Escolha Add schema (Adicionar esquema).
4. Digite um Schema name (Nome do esquema), consistindo em letras, números, hifens, sublinhados, cifrões ou marcas de hash. Esse nome não pode ser alterado.
5. Escolha o Registry (Registro) em que o esquema será armazenado no menu suspenso. O registro pai não pode ser alterado após a criação.
6. Deixe Data format (Formato de dados) como Apache Avro ou JSON. Esse formato se aplica a todas as versões desse esquema.
7. Escolha um Compatibility mode (Modo de compatibilidade).
   - Backward (Anterior) (recomendado): o receptor pode ler a versão atual e a anterior.
   - Backward all (Todas as anteriores): o receptor pode ler a versão atual e todas as anteriores.
   - Forward (Próxima): o remetente pode gravar a versão atual e a anterior.
   - Forward all (Todas as próximas): o remetente pode gravar a versão atual e todas as anteriores.
   - Full (Completo): combinação de Backward e Forward.
   - Full all (Completo total): combinação de Backward All e Forward All.
   - None (Nenhum): nenhuma verificação de compatibilidade é realizada.
   - Disabled (Desabilitado): impede qualquer versionamento para esse esquema.
8. Insira uma Description (Descrição) opcional de até 250 caracteres para o registro.

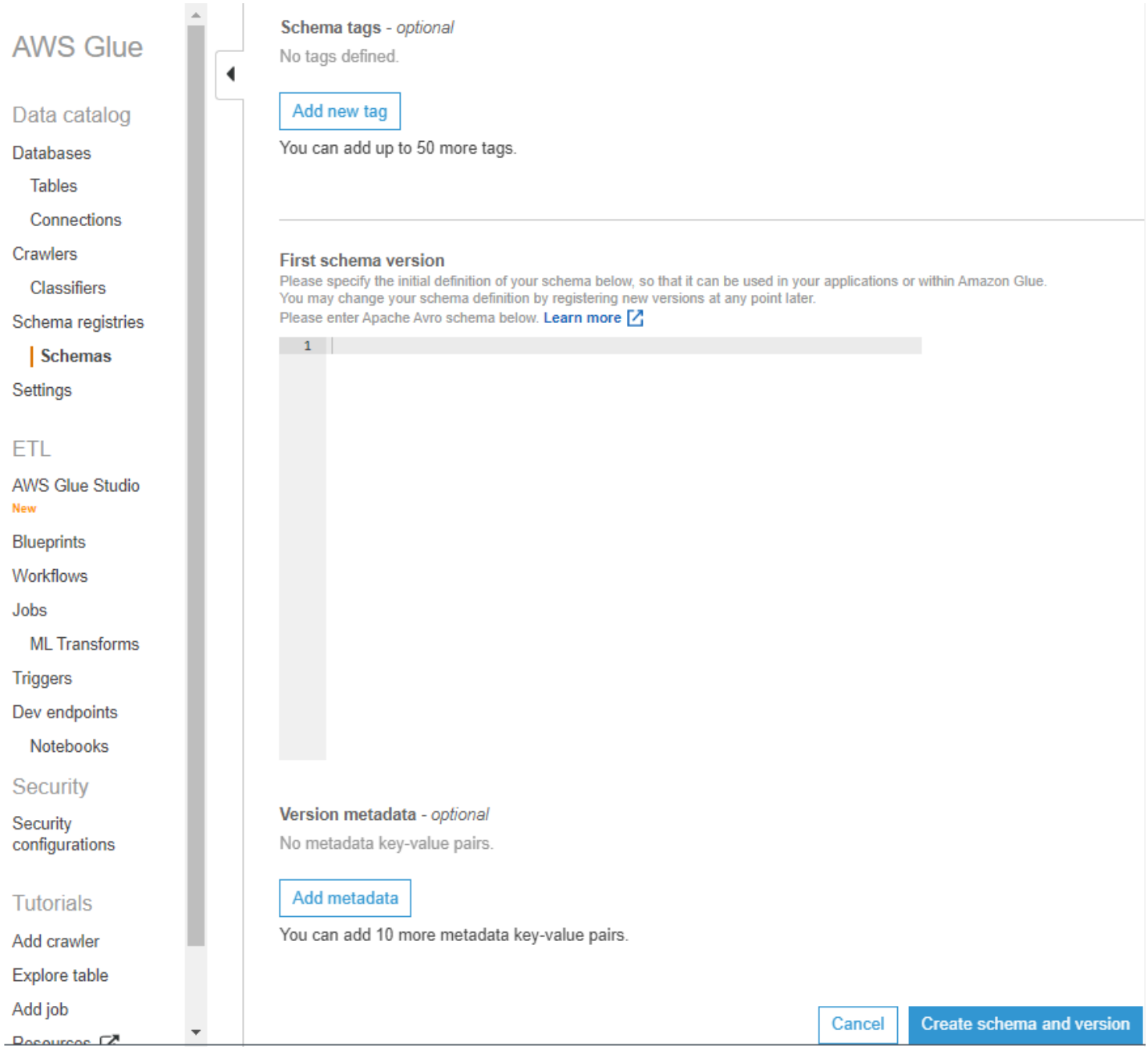


9. Opcionalmente, aplique uma ou mais tags ao esquema. Escolha Add new tag (Adicionar nova tag), especifique um Tag key (Chave de tag) e, opcionalmente, um Tag value (Valor da tag).
10. Na caixa First schema version (Primeira versão do esquema), insira ou cole o esquema inicial.

    Para o formato Avro, consulte Trabalhar com o formato de dados Avro

    Para o formato JSON, consulte Trabalhar com o formato de dados JSON

11. Opcionalmente, escolha Add metadata (Adicionar metadados) para adicionar metadados de versão para anotar ou classificar a versão do esquema.
12. Selecione Create schema and version (Criar esquema e versão).



O esquema é criado e aparece na lista em Schemas (Esquemas).

Trabalhar com o formato de dados Avro

O Avro fornece serviços de serialização e troca de dados. O Avro armazena a definição de dados no formato JSON facilitando a leitura e interpretação. Os dados em si são armazenados em formato binário.

Para obter informações sobre como definir um esquema do Apache Avro, consulte a especificação do Apache Avro.

Trabalhar com o formato de dados JSON

Os dados podem ser serializados com o formato JSON. O formato do esquema JSON define o padrão para o formato de esquema JSON.

Atualizar um esquema ou registro

Uma vez criados, você pode editar seus esquemas, versões de esquema ou registro.

Atualizar um registro

Você pode atualizar um registro usando as APIs do AWS Glue ou o console do AWS Glue. O nome de um registro existente não pode ser editado. Você pode editar a descrição de um registro.

APIs do AWS Glue

Para atualizar um registro existente, use a API UpdateRegistry ação (Python: update_registry).

Especifique uma estrutura `RegistryId` para indicar o registro que você deseja atualizar. Informe uma `Description` para alterar a descrição de um registro.

```
aws glue update-registry --description updatedDescription --registry-id
 RegistryArn="arn:aws:glue:us-east-2:901234567890:registry/registryName1"
```

Console do AWS Glue

Para atualizar um registro usando o console do AWS Glue:

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (Catálogo de dados), escolha Schema registries (Registros de esquemas).

3. Escolha um registro na lista de registros, marcando a caixa correspondente.
4. No menu Action (Ação), escolha Edit registry (Editar registro).

Atualizar um esquema

Você pode atualizar a descrição ou a configuração de compatibilidade de um esquema.

Para atualizar um esquema existente, use a API UpdateSchema ação (Python: update_schema).

Especifique uma estrutura `SchemaId` para indicar o esquema que você deseja atualizar. Um de `VersionNumber` ou `Compatibility` tem de ser fornecido.

Exemplo 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
```

Adicionar uma versão de esquema

Quando você adiciona uma versão de esquema, precisa comparar as versões para se certificar de que o novo esquema será aceito.

Para adicionar uma nova versão a um esquema existente, use a API RegisterSchemaVersion ação (Python: register_schema_version).

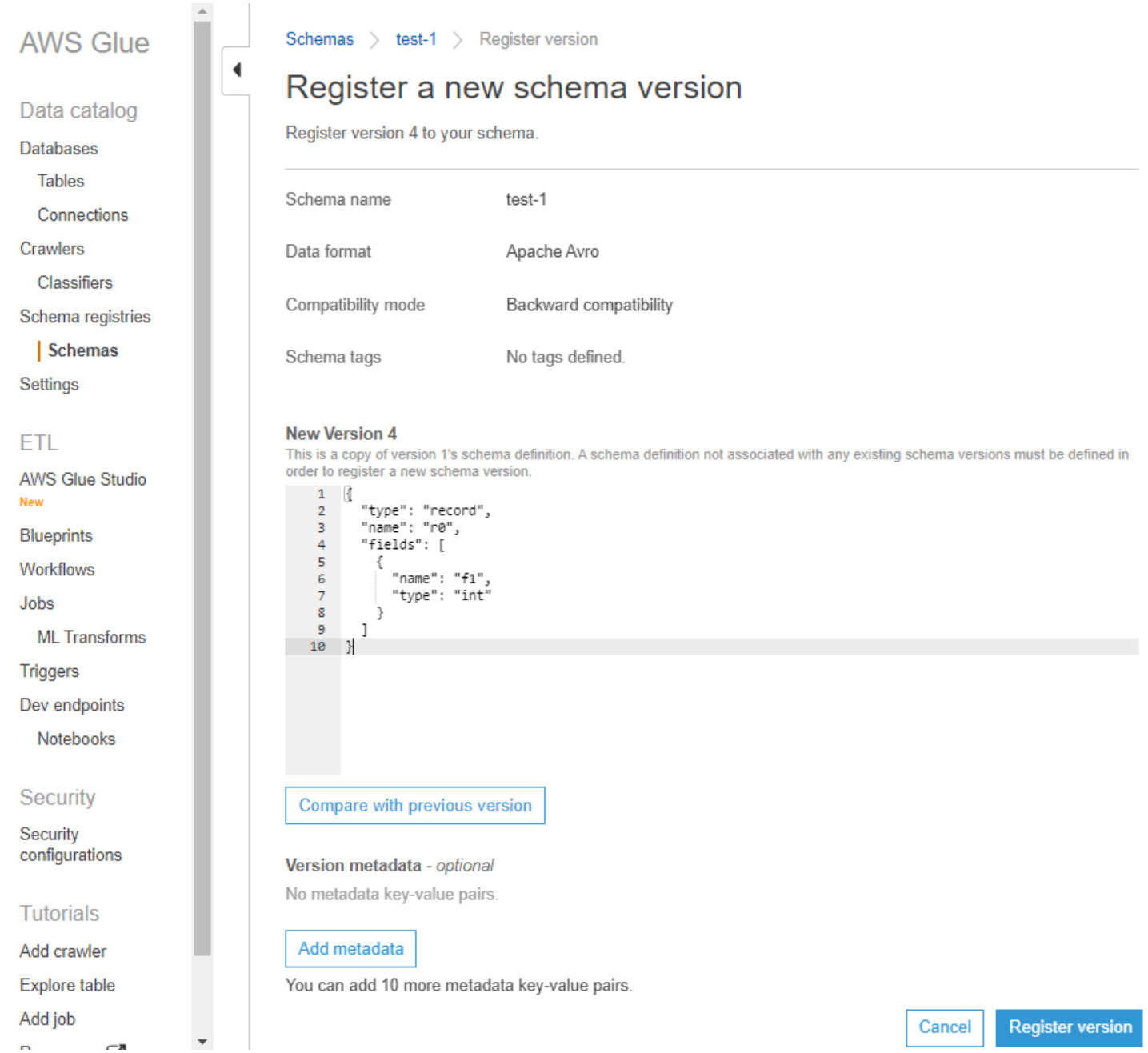
Especifique um estrutura `SchemaId` para indicar o esquema no qual você deseja adicionar uma versão e uma `SchemaDefinition` para definir o esquema.

Exemplo 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. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (Catálogo de dados), escolha Schema (Esquema).
3. Escolha o esquema na lista de esquemas, marcando a caixa correspondente.
4. Escolha um ou mais esquemas na lista, marcando as caixas.
5. No menu Action (Ação), escolha Register new version (Registrar nova versão).
6. Na caixa New version (Nova versão), insira ou cole seu novo esquema.
7. Escolha Compare with previous version (Comparar com a versão anterior) para ver as diferenças em relação à versão anterior do esquema.
8. Opcionalmente, escolha Add metadata (Adicionar metadados) para adicionar metadados de versão para anotar ou classificar a versão do esquema. Digite a Key (Chave) e o Value (Valor) opcional.
9. Escolha Register version (Registrar versão).



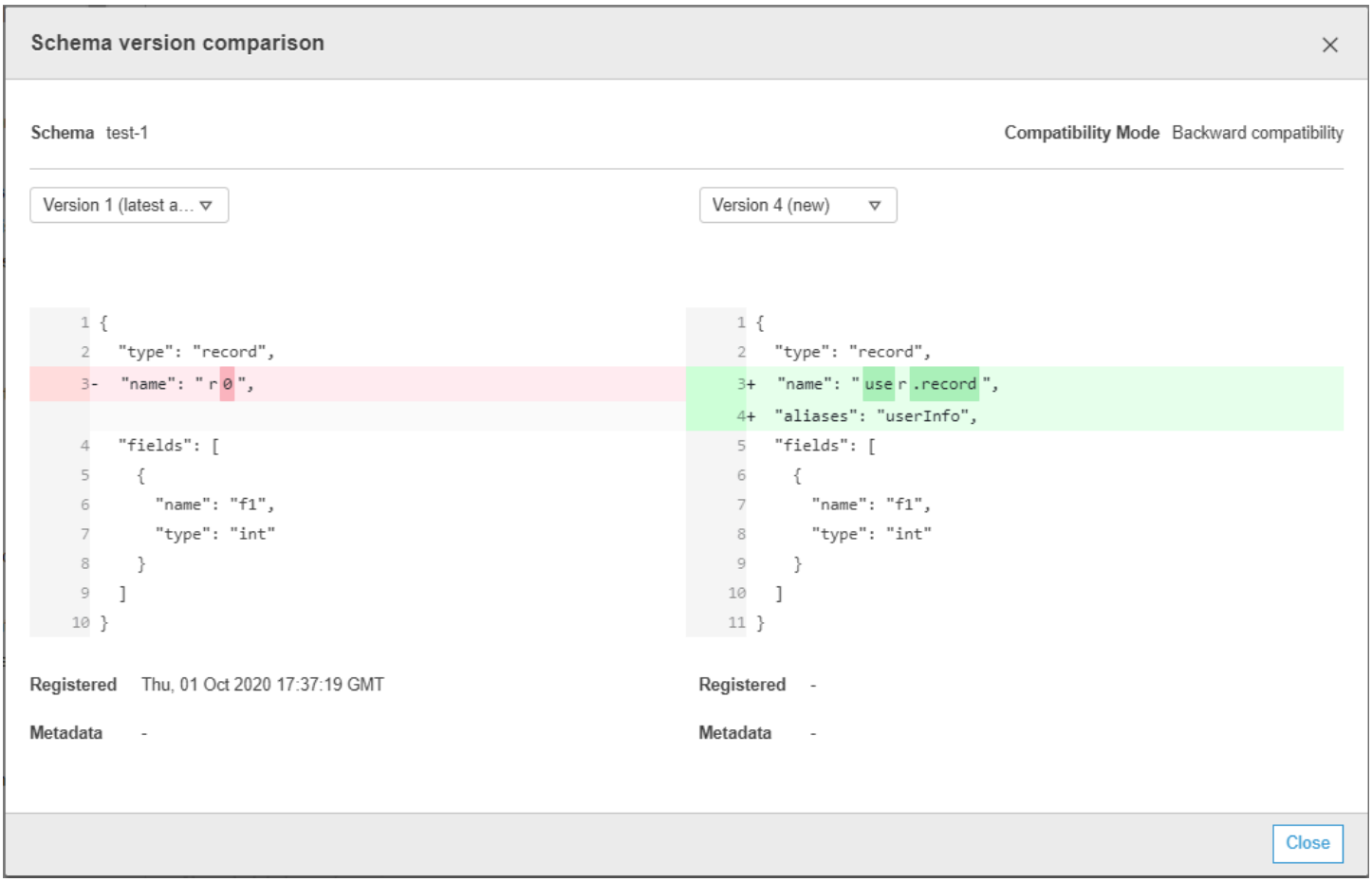
As versões dos esquemas aparecem na lista de versões. Se a versão alterou o modo de compatibilidade, ela será marcada como um ponto de verificação.

Exemplo de comparação de versões de esquema

Quando você escolher Compare with previous version (Comparar com a versão anterior), você verá a versão nova e a anterior exibidas juntas. As informações alteradas serão destacadas da seguinte forma:

- Amarelo: indica informações alteradas.
- Verde: indica o conteúdo adicionado na versão mais recente.
- Vermelho: indica o conteúdo removido da versão mais recente.

Você também pode comparar com versões anteriores.



Excluir um esquema ou registro

Excluir um esquema, uma versão de esquema ou um registro são ações permanentes que não podem ser desfeitas.

Excluir um esquema

Você pode querer excluir um esquema quando ele não for mais usado dentro de um registro usando o AWS Management Console ou a API DeleteSchema ação (Python: delete_schema).

Excluir um ou mais esquemas é uma ação permanente que não pode ser desfeita. Certifique-se de que o esquema ou esquemas não são mais necessários.

Para excluir um esquema do registro, chame a API DeleteSchema ação (Python: delete_schema), especificando a estrutura `SchemaId` para identificar o esquema.

Por exemplo:

```
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"
```

Console do AWS Glue

Para excluir um esquema com o console do AWS Glue:

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (Catálogo de dados), escolha Schema registries (Registros de esquemas).
3. Escolha o registro que contém o esquema na lista de registros.
4. Escolha um ou mais esquemas na lista, marcando as caixas.
5. No menu Actions (Ações), escolha Delete schema (Excluir esquema).
6. Insira o texto **`Delete`** no campo para confirmar a exclusão.
7. Escolha Excluir.

Os esquemas especificados são excluídos do registro.

Excluir uma versão de esquema

À medida que os esquemas se acumulam no registro, você pode querer excluir versões de esquema indesejadas usando o AWS Management Console ou a API DeleteSchemaVersions ação (Python: delete_schema_versions). Excluir uma ou mais versões de esquema é uma ação permanente que não pode ser desfeita. Certifique-se de que as versões de esquema não são mais necessárias.

Ao excluir versões de esquema, observe as seguintes restrições:

- Você não pode excluir uma versão marcada como ponto de verificação.

- O intervalo de versões contíguas não pode ser superior a 25.
- A versão mais recente do esquema não deve estar em estado pendente.

Especifique a estrutura `SchemaId` para identificar o esquema e especifique `Versions` como um intervalo de versões a serem excluídas. Para obter mais informações sobre como especificar uma versão ou intervalo de versões, consulte DeleteRegistry ação (Python: delete_registry). As versões de esquema especificadas são excluídas do registro.

Chamar a API ListSchemaVersions ação (Python: list_schema_versions) após essa chamada listará o status das versões excluídas.

Por exemplo:

```
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. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (Catálogo de dados), escolha Schema registries (Registros de esquemas).
3. Escolha o registro que contém o esquema na lista de registros.
4. Escolha um ou mais esquemas na lista, marcando as caixas.
5. No menu Actions (Ações), escolha Delete schema (Excluir esquema).
6. Insira o texto **`Delete`** no campo para confirmar a exclusão.
7. Escolha Excluir.

As versões de esquema especificadas são excluídas do registro.

Excluir um registro

Você pode querer excluir um registro quando os esquemas que ele contém não devem mais ser organizados nele. Você precisará reatribuir esses esquemas a outro registro.

Excluir um ou mais registros é uma ação permanente que não pode ser desfeita. Certifique-se de que o registro ou registros não são mais necessários.

O registro padrão pode ser excluído usando a AWS CLI.

API do AWS Glue

Para excluir todo o registro, incluindo o esquema e todas as suas versões, chame a API DeleteRegistry ação (Python: delete_registry). Especifique uma estrutura `RegistryId` para identificar o registro.

Por exemplo:

```
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 obter o status da operação de exclusão, é possível chamar a API `GetRegistry` após a chamada assíncrona.

Console do AWS Glue

Para excluir um registro do console do AWS Glue:

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (Catálogo de dados), escolha Schema registries (Registros de esquemas).
3. Escolha um registro na lista, marcando uma caixa.
4. No menu Action (Ação), escolha Delete registry (Excluir registro).
5. Insira o texto **`Delete`** no campo para confirmar a exclusão.
6. Escolha Excluir.

Os registros que você selecionou são excluídos do AWS Glue.

Exemplos do IAM para serializadores

> **Note**
>
> As políticas gerenciadas pela AWS concedem as permissões necessárias para casos de uso comuns. Para obter informações sobre como usar políticas gerenciadas para gerenciar o registro do esquema, consulte AWS políticas gerenciadas (predefinidas) para AWS Glue.

Para serializadores, você deve criar uma política mínima semelhante à abaixo, para lhe dar a capacidade de encontrar o `schemaVersionId` para uma determinada definição de esquema. Observe que você deve ter permissões de leitura no registro para ler os esquemas no registro. Você pode limitar os registros que podem ser lidos usando a cláusula `Resource`.

Exemplo 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"
                     ]
}
```

Além disso, você também pode permitir que os produtores criem novos esquemas e versões ao incluir os seguintes métodos extras. Observe que você deve ser capaz de inspecionar o registro para adicionar/remover/evoluir os esquemas dentro dele. Você pode limitar os registros que podem ser inspecionados usando a cláusula `Resource`.

Exemplo 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"
                ]
}
```

Exemplos do IAM para desserializadores

Para desserializadores (lado do consumidor), você deve criar uma política semelhante à abaixo para permitir que o desserializador busque o esquema do registro de esquemas para desserialização. Observe que você deve ser capaz de inspecionar o registro a fim de buscar os esquemas dentro dele.

Exemplo de código 15:

```
{
    "Sid" : "GetSchemaVersion",
    "Effect" : "Allow",
    "Action" :
 [
        "glue:GetSchemaVersion"
    ],
    "Resource" : ["*"]
}
```

Conectividade privada usando AWS PrivateLink

Você pode usar o AWS PrivateLink para conectar a VPC do produtor de dados ao AWS Glue definindo um endpoint da VPC de interface para o AWS Glue. Quando você usa um endpoint da VPC de interface, a comunicação entre sua VPC e o AWS Glue é realizada inteiramente dentro da rede da AWS. Para obter mais informações, consulte Usar o AWS Glue com endpoints da VPC.

Acessar métricas do Amazon CloudWatch

As métricas do Amazon CloudWatch estão disponíveis como parte do nível gratuito do CloudWatch. Você também pode visualizar essas métricas no console do CloudWatch. As métricas de nível de API incluem CreateSchema (sucesso e latência), GetSchemaByDefinition (sucesso e latência), GetSchemaVersion (sucesso e latência), RegisterSchemaVersion (sucesso e latência) e PutSchemaVersionMetadata (sucesso e latência). As métricas de nível de recurso incluem Registry.ThrottledByLimit, SchemaVersion.ThrottledByLimit e SchemaVersion.Size.

Exemplo do modelo do AWS CloudFormation para o registro de esquemas

Veja a seguir um modelo de exemplo para criar recursos do registro de esquemas no AWS CloudFormation. Para criar essa pilha em sua conta, copie o modelo acima em um arquivo `SampleTemplate.yaml` e execute o seguinte comando:

```
aws cloudformation create-stack --stack-name ABCSchemaRegistryStack --template-body
 "'cat SampleTemplate.yaml'"
```

Este exemplo usa `AWS::Glue::Registry` para criar um registro, `AWS::Glue::Schema` para criar um esquema, `AWS::Glue::SchemaVersion` para criar uma versão de esquema e `AWS::Glue::SchemaVersionMetadata` para preencher os metadados da versão do 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"
```

Integração com o registro de esquemas do AWS Glue

Estas seções descrevem integrações com o registro de esquemas do AWS Glue. Os exemplos nesta seção mostram um esquema com formato de dados AVRO. Para obter mais exemplos, incluindo esquemas com formato de dados JSON, consulte os testes de integração e as informações no arquivo ReadMe no repositório de código aberto do registro de esquemas do AWS Glue.

Tópicos

- Caso de uso: conectar registro de esquemas ao Amazon MSK ou Apache Kafka
- Caso de uso: integração do Amazon Kinesis Data Streams ao registro de esquemas do AWS Glue
- Caso de uso: Amazon Managed Service for Apache Flink
- Caso de uso: integração com o AWS Lambda

- Caso de uso: AWS Glue Data Catalog
- Caso de uso: transmissão no AWS Glue
- Caso de uso: Apache Kafka Streams
- Caso de uso: Apache Kafka Connect

Caso de uso: conectar registro de esquemas ao Amazon MSK ou Apache Kafka

Vamos supor que você está gravando dados em um tópico do Apache Kafka. Você pode seguir estas etapas para começar.

1. Crie um cluster do Amazon Managed Streaming for Apache Kafka (Amazon MSK) ou do Apache Kafka com pelo menos um tópico. Se estiver criando um cluster do Amazon MSK, você pode usar o AWS Management Console. Para obter mais informações: Conceitos básicos do uso do Amazon MSK no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.
2. Siga o passo Instalar as bibliotecas SerDe acima.
3. Para criar registros de esquemas, esquemas ou versões de esquema, siga as instruções na seção Conceitos básicos do registro de esquemas deste documento.
4. Inicie seus produtores e consumidores para usar o registro de esquemas para gravar e ler registros de/para o tópico do Amazon MSK ou Apache Kafka. Um exemplo de código de produtor e consumidor pode ser encontrado no arquivo ReadMe das bibliotecas Serde. A biblioteca do registro de esquemas no produtor serializará automaticamente o registro e decorará o registro com um ID de versão do esquema.
5. Se o esquema desse registro tiver sido inserido, ou se o registro automático estiver ativado, o esquema será registrado no registro do esquemas.
6. O consumidor lendo do tópico do Amazon MSK ou Apache Kafka, usando a biblioteca do registro de esquemas do AWS Glue, pesquisará automaticamente o esquema no registro do esquemas.

Caso de uso: integração do Amazon Kinesis Data Streams ao registro de esquemas do AWS Glue

Essa integração requer que você tenha um fluxo de dados existente do Amazon Kinesis. Para obter mais informações, consulte Conceitos básicos do Amazon Kinesis Data Streams no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Há duas maneiras de interagir com os dados em um fluxo de dados do Kinesis.

- Por meio das bibliotecas do Kinesis Producer Library (KPL) e Kinesis Client Library (KCL) em Java. Suporte a várias linguagens não é fornecido.
- Por maio das APIs `PutRecords`, `PutRecord` e `GetRecords` do Kinesis Data Streams disponíveis no AWS SDK for Java.

Se você usa atualmente as bibliotecas KPL/KCL, recomendamos continuar usando esse método. Há versões atualizadas da KCL e KPL com o registro de esquemas integrado, como mostrado nos exemplos. Caso contrário, você pode usar o código de exemplo para utilizar o registro do esquemas do AWS Glue se estiver usando as APIs do KDS diretamente.

A integração do registro de esquemas só está disponível com a KPL v0.14.2 ou posterior e com a KCL v2.3 ou posterior. A integração do registro de esquemas com JSON só está disponível com a KPL v0.14.8 ou posterior e com a KCL v2.3.6 ou posterior.

Interagir com dados usando o Kinesis SDK V2

Esta seção descreve a interação com o Kinesis usando o Kinesis SDK 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);
}
```

Interagir com dados usando as bibliotecas KPL/KCL

Esta seção descreve a integração do Kinesis Data Streams com o registro de esquemas usando as bibliotecas KPL/KCL. Para obter mais informações sobre o uso da KPL/KCL, consulte Desenvolver produtores usando a biblioteca de produtor do Amazon Kinesis no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Configurar o registro de esquemas na KPL

1. Configure a definição do esquema para os dados, o formato de dados e o nome do esquema criados no registro de esquemas do AWS Glue.
2. Opcionalmente, configure o objeto `GlueSchemaRegistryConfiguration`.
3. Transmita o objeto de esquema para `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());
 }
```

Configurar a biblioteca do cliente Kinesis

Você desenvolverá seu consumidor da biblioteca do cliente Kinesis em Java. Para obter mais informações, consulte Desenvolver um consumidor da biblioteca do cliente Kinesis em Java no Guia do desenvolvedor do Amazon Kinesis Data Streams.

1. Crie uma instância de `GlueSchemaRegistryDeserializer` transmitindo um objeto `GlueSchemaRegistryConfiguration`.
2. Transmita o `GlueSchemaRegistryDeserializer` para `retrievalConfig.glueSchemaRegistryDeserializer`.
3. Acesse o esquema de mensagens recebidas chamando `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);
 }
```

Interagir com dados usando as APIs do Kinesis Data Streams

Esta seção descreve a integração do Kinesis Data Streams com o registro de esquemas usando as APIs do Kinesis Data Streams.

1. Atualize estas dependências do 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. No produtor, adicione as informações de cabeçalho do esquema usando a API `PutRecords` ou `PutRecord` no 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. No produtor, use a API `PutRecords` ou `PutRecord` para colocar o registro no fluxo de dados.
4. No consumidor, remova o registro do esquema do cabeçalho e serialize um registro do esquema 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);
        }
```

Interagir com dados usando as APIs do Kinesis Data Streams

O código de exemplo a seguir usa as APIs `PutRecords` e `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 for Apache Flink

O Apache Flink é um framework de código aberto popular e mecanismo de processamento distribuído para computações com estado sobre fluxos de dados vinculados e não vinculados. O Amazon Managed Service for Apache Flink é um serviço totalmente gerenciado da AWS que permite a criação e o gerenciamento de aplicações do Apache Flink para processar dados de transmissão.

O Apache Flink de código aberto fornece uma série de fontes e coletores. Por exemplo, origens de dados predefinidas incluem leitura de arquivos, diretórios e soquetes e ingestão de dados de coleções e iteradores. Os conectores Apache Flink DataStream fornecem código para o Apache Flink para realizar a interface com vários sistemas de terceiros, como Apache Kafka ou Kinesis, como fontes e/ou coletores.

Para obter mais informações, consulte o Guia do desenvolvedor do Amazon Kinesis Data Analytics.

Conector Apache Flink Kafka

O Apache Flink fornece um conector de fluxo de dados Apache Kafka para leitura e gravação de dados em tópicos do Kafka com garantias do tipo exatamente uma. Consumidor Kafka do Flink,

`FlinkKafkaConsumer`, fornece acesso à leitura de um ou mais tópicos do Kafka. O produtor Kafka do Apache Flink, `FlinkKafkaProducer`, permite gravar uma transmissão de registros para um ou mais tópicos do Kafka. Para obter mais informações, consulte Conector do Apache Kafka.

Conector de fluxos do Kinesis do Apache Flink

O conector de transmissão de dados do Kinesis fornece acesso aos Amazon Kinesis Data Streams. O `FlinkKinesisConsumer` é uma origem de dados de transmissão paralelo do tipo exatamente uma vez que assina várias transmissões do Kinesis dentro da mesma região de produto da AWS e que pode manipular de forma transparente a refragmentação de transmissões enquanto o trabalho é executado. Cada subtarefa do consumidor é responsável por buscar registros de dados de vários fragmentos do Kinesis. O número de fragmentos obtidos por cada subtarefa será alterado à medida que os fragmentos forem fechados e criados pelo Kinesis. O `FlinkKinesisProducer` usa a Kinesis Producer Library (KPL) para colocar dados de uma transmissão do Apache Flink em uma transmissão do Kinesis. Para obter mais informações, consulte Conector do Amazon Kinesis Streams.

Para obter mais informações, consulte o repositório do GitHub de registro do AWS Glue.

Integração com o Apache Flink

A biblioteca SerDes fornecida com o registro de esquemas se integra com o Apache Flink. Para trabalhar com o Apache Flink, é necessário implementar as interfaces `SerializationSchema` e `DeserializationSchema` chamadas `GlueSchemaRegistryAvroSerializationSchema` e `GlueSchemaRegistryAvroDeserializationSchema`, que você pode vincular aos conectores Apache Flink.

Adicionar uma dependência do registro do esquemas do AWS Glue na aplicação do Apache Flink

Para configurar as dependências de integração para o registro de esquemas do AWS Glue na aplicação do Apache Flink:

1. Adicione a dependência ao arquivo `pom.xml`.

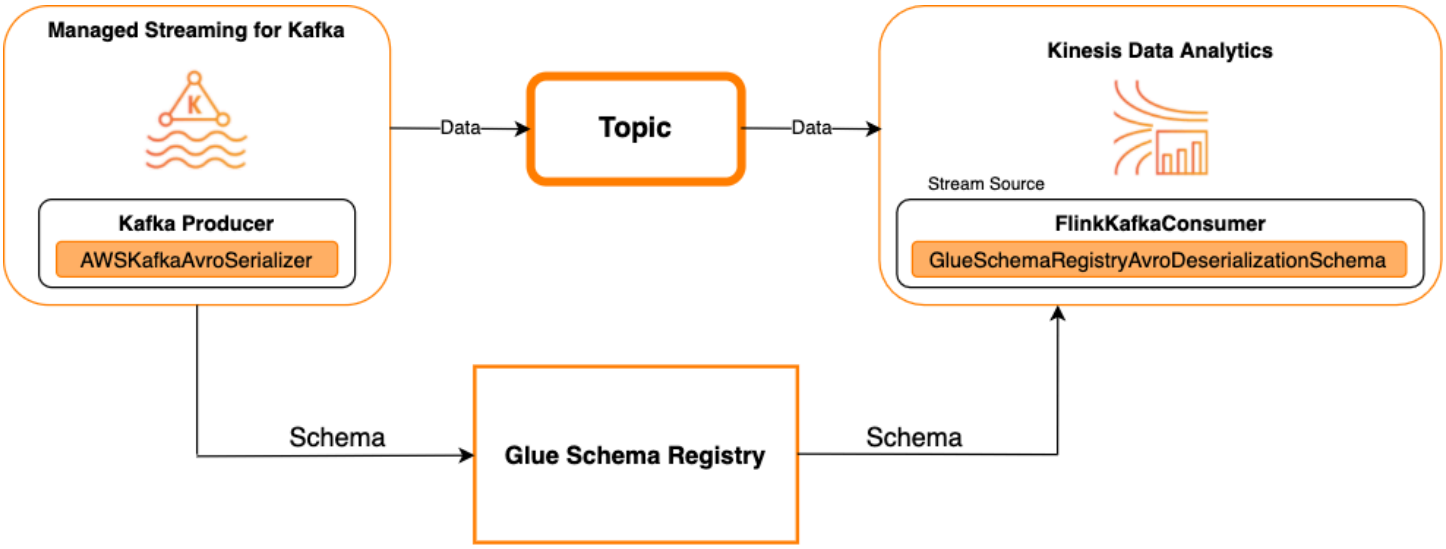
```
<dependency>
    <groupId>software.amazon.glue</groupId>
    <artifactId>schema-registry-flink-serde</artifactId>
    <version>1.0.0</version>
</dependency>
```

Integração do Kafka ou do Amazon MSK com o Apache Flink

Você pode usar o Managed Service for Apache Flink, com o Kafka como fonte ou com o Kafka como coletor.

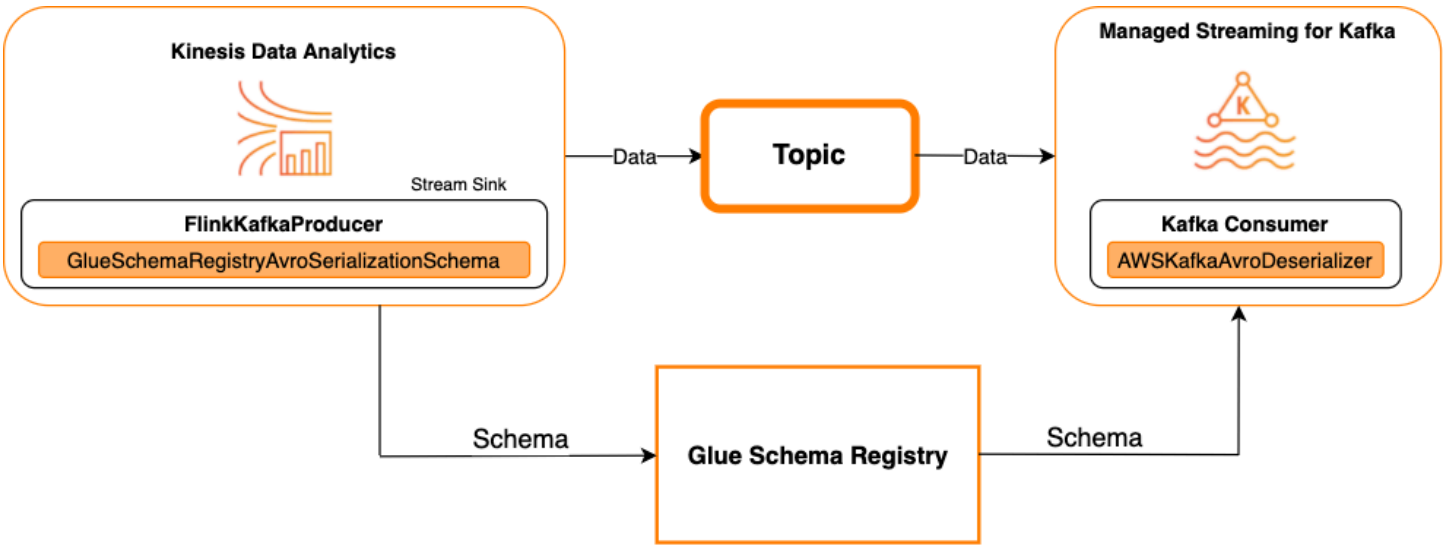
Kafka como uma fonte

O diagrama a seguir mostra a integração do Kinesis Data Streams com o Managed Service for Apache Flink com o Kafka como uma fonte.



Kafka como um coletor

O diagrama a seguir mostra a integração do Kinesis Data Streams com o Managed Service for Apache Flink com o Kafka como uma coletor.



Para integrar o Kafka (ou o Amazon MSK) com o Managed Service for Apache Flink com o Kafka como uma fonte ou como um coletor, faça as alterações de código abaixo. Adicione os blocos de código em negrito ao seu respectivo código nas seções análogas.

Se o Kafka for a fonte, use o código de desserialização (bloco 2). Se o Kafka for o coletor, use o código serializador (bloco 3).

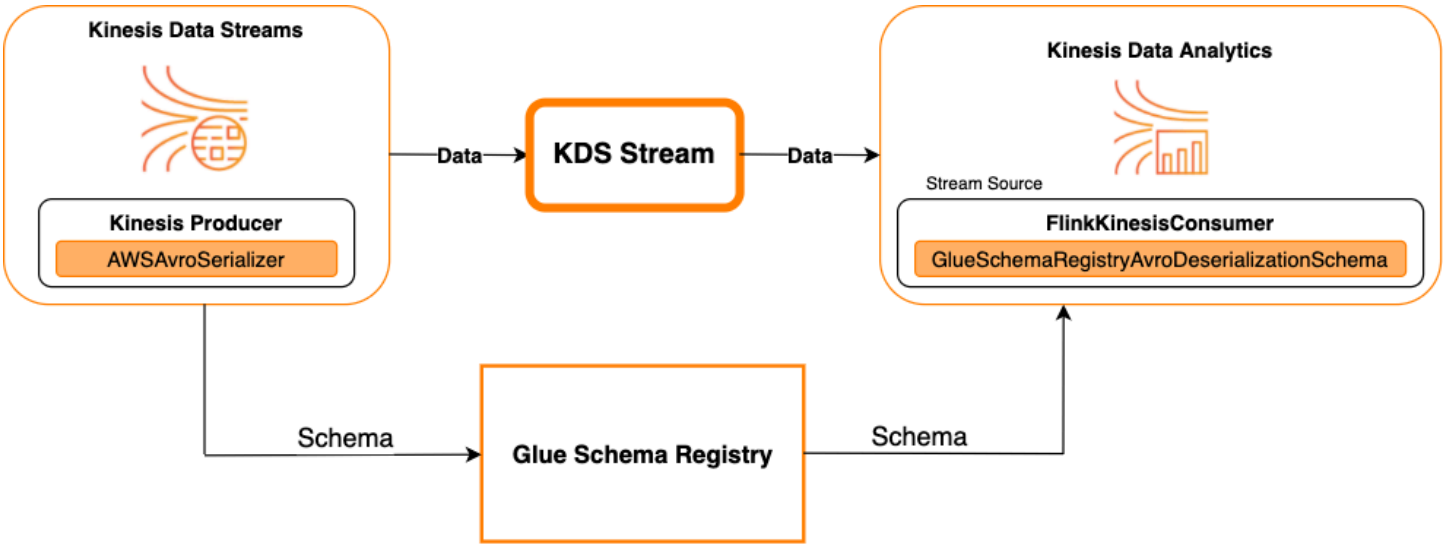
```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

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();
```

Integração do Kinesis Data Streams com o Apache Flink

Você pode usar o Managed Service for Apache Flink com o Kinesis Data Streams como fonte ou como coletor.

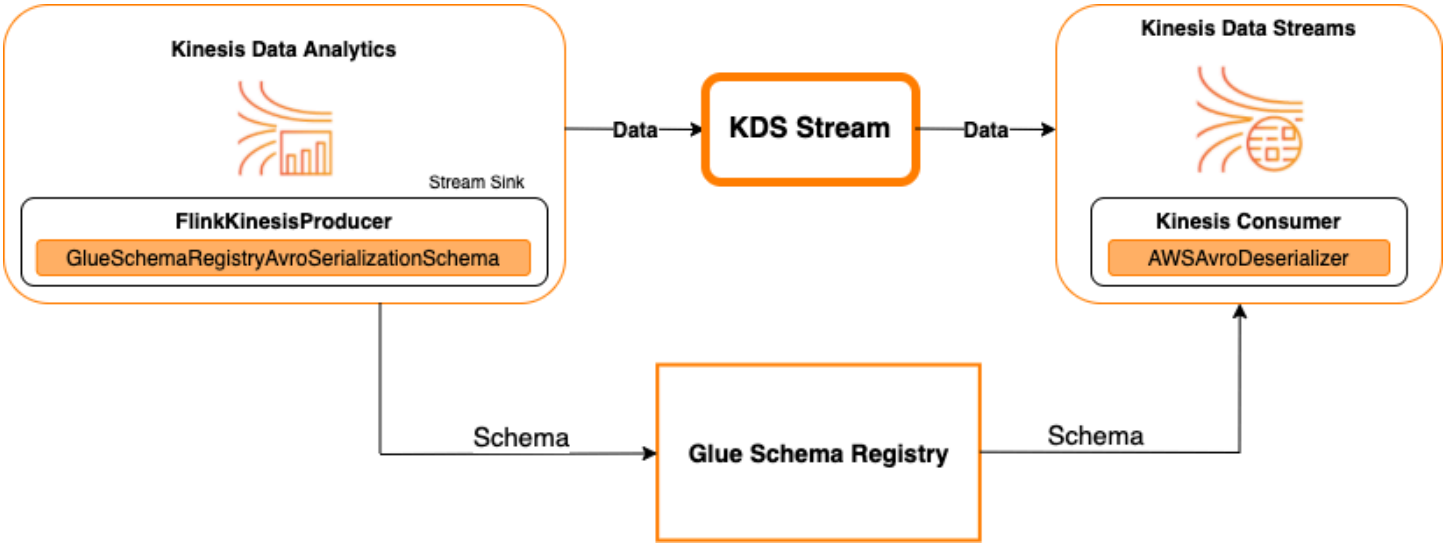
Kinesis Data Streams como uma fonte

O diagrama a seguir mostra a integração do Kinesis Data Streams com o Managed Service for Apache Flink com o Kinesis Data Streams como uma fonte.



Kinesis Data Streams como um coletor

O diagrama a seguir mostra a integração do Kinesis Data Streams com o Managed Service for Apache Flink com o Kinesis Data Streams como uma coletor.



Para integrar o Kinesis Data Streams com o Managed Service for Apache Flink com o Kinesis Data Streams como uma origem ou como um coletor, faça as alterações de código abaixo. Adicione os blocos de código em negrito ao seu respectivo código nas seções análogas.

Se o Kinesis Data Streams for a fonte, use o código de desserialização (bloco 2). Se o Kinesis Data Streams for o coletor, use o código de serialização (bloco 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: integração com o AWS Lambda

Para usar uma função do AWS Lambda como um consumidor do Apache Kafka/Amazon MSK e desserializar mensagens codificadas em Avro usando o registro de esquemas do AWS Glue, acesse a página do MSK Labs.

Caso de uso: AWS Glue Data Catalog

As tabelas do AWS Glue suportam esquemas que você pode especificar manualmente ou por referência aoregistro de esquemas do AWS Glue. O registro de esquemas se integra ao Data Catalog para permitir que você opcionalmente use esquemas armazenados no registro de esquemas ao criar ou atualizar tabelas ou partições do AWS Glue no Data Catalog. Para identificar uma definição de esquema no registro de esquemas, no mínimo, você precisa saber o ARN do esquema do qual ele faz parte. Uma versão de esquema que contém uma definição de esquema pode ser referenciada por seu UUID ou número de versão. Há sempre uma versão de esquema, a versão "mais recente", que pode ser pesquisada sem saber seu número de versão ou UUID.

Ao chamar as operações `CreateTable` ou `UpdateTable`, você transmitirá uma estrutura `TableInput` que contém um `StorageDescriptor`, que pode ter uma `SchemaReference` para um esquema existente no registro de esquemas. Da mesma forma, quando você chama as APIs `GetTable` ou `GetPartition`, a resposta pode conter o esquema e a `SchemaReference`. Quando uma tabela ou partição é criada usando referências de esquema, o Data Catalog tenta buscar o esquema para essa referência. Caso não seja possível localizar o esquema no registro de esquemas, ele retorna um esquema vazio na reposta de `GetTable`. Caso contrário, a resposta possui o esquema e a referência do esquema.

Também é possível executar as ações no console do AWS Glue.

Para realizar essas operações e criar, atualizar ou exibir informações de esquema, você deve conceder ao usuário que faz a chamada um perfil do IAM que forneça permissões para a API `GetSchemaVersion`.

Adicionar uma tabela ou atualizar o esquema de uma tabela

A adição de uma nova tabela a partir de um esquema existente vincula a tabela a uma versão específica do esquema. Depois que as novas versões de esquema forem registradas, você poderá atualizar essa definição de tabela na página View table (Exibir tabela) no console do AWS Glue ou usando a API UpdateTable ação (Python: update_table).

Adicionar uma tabela de um esquema existente

É possível criar uma tabela do AWS Glue a partir de uma versão de esquema no registro usando o console do AWS Glue ou a API `CreateTable`.

API do AWS Glue

Ao chamar a API `CreateTable`, você transmitirá uma `TableInput` que contém um `StorageDescriptor` que, por sua vez, contém uma `SchemaReference` para um esquema existente no registro de esquemas.

Console do AWS Glue

Para criar uma tabela no console do AWS Glue:

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (Catálogo de dados), escolha Tables (Tabelas).
3. No menu Add Tables (Adicionar tabelas), escolha Add table from existing schema (Adicionar tabela do esquema existente).
4. Configure as propriedades da tabela e o datastore de acordo com o guia do desenvolvedor do AWS Glue.
5. Na página Choose a Glue schema (Escolher um esquema do Glue), selecione o Registry (Registro) em que o esquema reside.
6. Escolha o Schema name (Nome do esquema) e selecione a Version (Versão) do esquema a ser aplicada.
7. Revise a pré-visualização do esquema e escolha Next (Próximo).
8. Revise e crie a tabela.

O esquema e a versão aplicada à tabela são exibidos na coluna Glue schema (Esquema do Glue) na lista de tabelas. Você pode exibir a tabela para ver mais detalhes.

Atualizar o esquema de uma tabela

Quando uma nova versão de esquema se torna disponível, você pode querer atualizar o esquema de uma tabela usando a API UpdateTable ação (Python: update_table) ou o console do AWS Glue.

> **Important**
>
> Ao atualizar o esquema de uma tabela existente que tenha um esquema do AWS Glue especificado manualmente, o novo esquema referenciado no registro de esquemas pode ser incompatível. Isso pode resultar na falha de seus trabalhos.

API do AWS Glue

Ao chamar a API `UpdateTable`, você transmitirá uma `TableInput` que contém um `StorageDescriptor` que, por sua vez, contém uma `SchemaReference` para um esquema existente no registro de esquemas.

Console do AWS Glue

Para atualizar o esquema de uma tabela a partir do console do AWS Glue:

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (Catálogo de dados), escolha Tables (Tabelas).
3. Exiba a tabela na lista de tabelas.
4. Clique em Update schema (Atualizar esquema) na caixa que informa sobre uma nova versão.
5. Revise as diferenças entre o esquema atual e o novo.
6. Escolha Show all schema differences (Mostrar todas as diferenças do esquema) para ver mais detalhes.
7. Escolha Save table (Salvar tabela) para aceitar a nova versão.

Caso de uso: transmissão no AWS Glue

A transmissão no AWS Glue consome dados de fontes de transmissão e executa operações de ETL antes de gravar em um coletor de saída. É possível especificar a fonte de entrada da transmissão usando uma tabela de dados ou diretamente, especificando a configuração da fonte.

A transmissão no AWS Glue oferece suporte a uma tabela do tipo Data Catalog para a fonte de transmissão criada com o esquema presente no AWS Glue Schema Registry. Você pode criar um esquema no AWS Glue Schema Registry e criar uma tabela do AWS Glue com uma fonte de transmissão usando esse esquema. Essa tabela do AWS Glue pode ser usada como entrada para um trabalho de transmissão do AWS Glue para desserialização de dados no fluxo de entrada.

É importante ressaltar que quando o esquema no AWS Glue Schema Registry muda, você precisa reiniciar o trabalho de transmissão do AWS Glue para refletir as alterações no esquema.

Caso de uso: Apache Kafka Streams

A API Apache Kafka Streams é uma biblioteca cliente para processamento e análise de dados armazenados no Apache Kafka. Esta seção descreve a integração do Apache Kafka Streams com o registro de esquemas do AWS Glue, que permite que você gerencie e imponha esquemas em suas

aplicações de transmissão de dados. Para obter mais informações sobre o Apache Kafka Streams, consulte Apache Kafka Streams.

Integração com as bibliotecas SerDes

Existe uma classe `GlueSchemaRegistryKafkaStreamsSerde` com a qual você pode configurar uma aplicação do Streams.

Código de exemplo da aplicação Kafka Streams

Para usar o registro de esquemas do AWS Glue dentro de uma aplicação Apache Kafka Streams:

1. Configure a aplicação 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. Crie uma transmissão a partir do tópico avro-input.

```
StreamsBuilder builder = new StreamsBuilder();
final KStream<String, GenericRecord> source = builder.stream("avro-input");
```

3. Processe os registros de dados (o exemplo filtra aqueles registros cujo valor de favorite_color é rosa ou onde o valor de quantidade é 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. Grave os resultados de volta no tópico avro-output.

```
result.to("avro-output");
```

5. Inicie a aplicação Apache Kafka Streams.

```
KafkaStreams streams = new KafkaStreams(builder.build(), props);
streams.start();
```

Resultados de implantação

Esses resultados mostram o processo de filtragem de registros que foram filtrados na etapa 3 como um favorite_color “rosa” ou o valor “15,0”.

Registros antes da filtragem:

```
{"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}
```

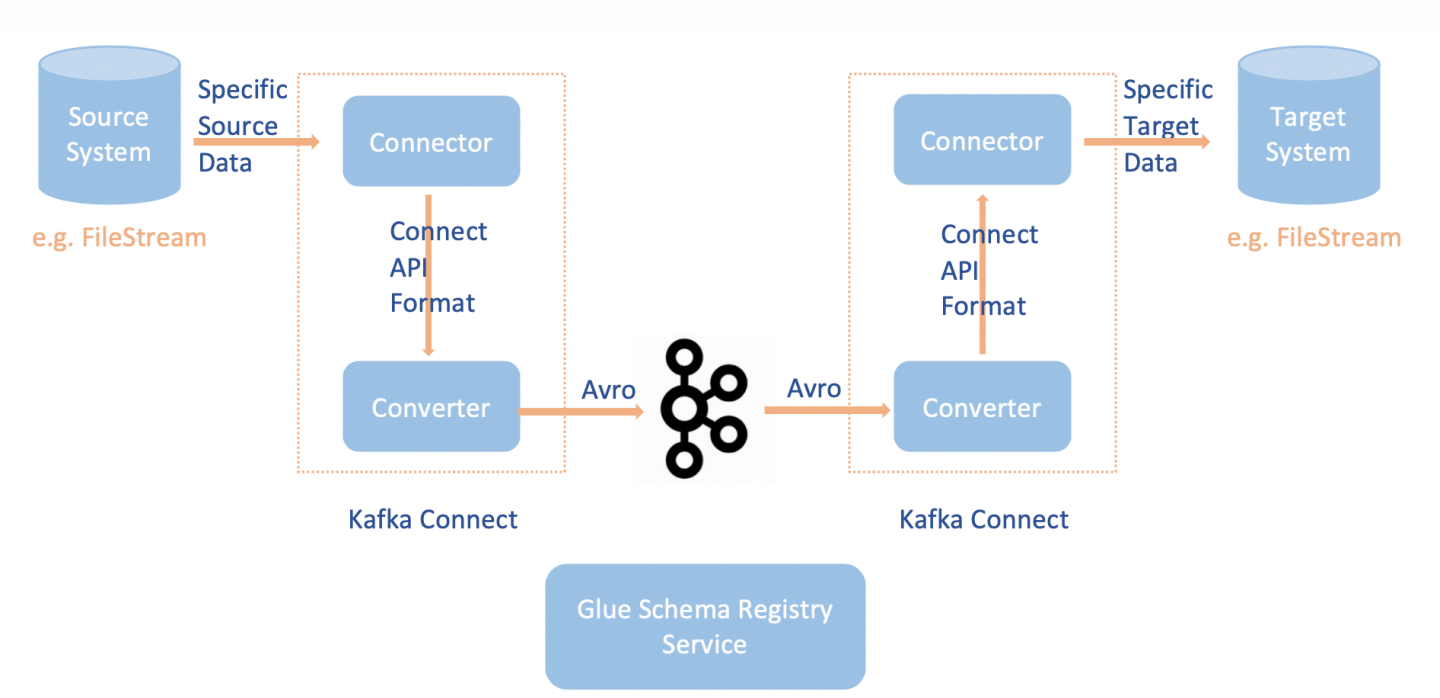
Registros após a filtragem:

```
{"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

A integração do Apache Kafka Connect com o registro de esquemas do AWS Glue permite que você obtenha informações de esquema dos conectores. Os conversores Apache Kafka especificam o formato dos dados dentro do Apache Kafka e como traduzi-los em dados do Apache Kafka Connect. Cada usuário do Apache Kafka Connect precisará configurar esses conversores com base no formato em que deseja que seus dados sejam carregados do, ou armazenados no, Apache Kafka. Desta forma, você pode definir seus próprios conversores para traduzir os dados do Apache Kafka Connect para o tipo usado no registro de esquemas do AWS Glue (por exemplo: Avro) e utilizar nosso serializador para registrar seu esquema e fazer a serialização. Em seguida, os conversores também são capazes de usar nosso desserializador para desserializar os dados recebidos do Apache Kafka e convertê-los de volta em dados do Apache Kafka Connect. Um exemplo de diagrama de fluxo de trabalho é dado abaixo.



1. Instale o projeto `aws-glue-schema-registry` clonando o [repositório do Github para o registro de esquemas do AWS Glue].

```
git clone git@github.com:awslabs/aws-glue-schema-registry.git
cd aws-glue-schema-registry
mvn clean install
mvn dependency:copy-dependencies
```

2. Se você planeja usar o Apache Kafka Connect no modo autônomo, atualize connect-standalone.properties usando as instruções abaixo para esta etapa. Se você planeja usar o Apache Kafka Connect no modo distribuído, atualize connect-avro-distributed.properties usando as mesmas instruções.

   a. Adicione essas propriedades também ao arquivo de propriedades de conexão do 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. Adicione o comando abaixo à seção Launch mode (Modo de execução) em kafka-run-class.sh:

   ```
   -cp $CLASSPATH:"<your AWS GlueSchema Registry base directory>/target/dependency/*"
   ```

3. Adicione o comando abaixo à seção Launch mode (Modo de execução) em kafka-run-class.sh

   ```
   -cp $CLASSPATH:"<your AWS GlueSchema Registry base directory>/target/dependency/*"
   ```

   A aparência deve ser semelhante a esta:

   ```
   # 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. Se estiver usando bash, execute os comandos abaixo para configurar seu CLASSPATH em seu bash_profile. Para qualquer outro shell, atualize o ambiente de acordo.

```
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) Se você quiser testar com uma fonte de arquivo simples, clone o conector da fonte do arquivo.

```
git clone https://github.com/mmolimar/kafka-connect-fs.git
cd kafka-connect-fs/
```

   a. Na configuração do conector da fonte, edite o formato de dados para Avro, leitor de arquivos para `AvroFileReader` e atualize um objeto Avro de exemplo a partir do caminho do arquivo em que você está lendo. Por exemplo:

```
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 o conector da fonte.

```
mvn clean package
echo "export CLASSPATH=\$CLASSPATH:\"\$(find target/ -type f -name '*.jar'| grep
 '\-package' | tr '\n' ':')\"" >>~/.bash_profile
source ~/.bash_profile
```

   c. Atualize as propriedades do coletor em *<your Apache Kafka installation directory>*`/config/connect-file-sink.properties`, atualize o nome do tópico e o nome do arquivo de saída.

```
file=<output file full path>
topics=<my topic>
```

6. Inicie o conector da fonte (neste exemplo, é um conector de fonte de arquivo).

```
$KAFKA_HOME/bin/connect-standalone.sh $KAFKA_HOME/config/connect-
standalone.properties config/kafka-connect-fs.properties
```

7. Execute o conector do coletor (neste exemplo, é um conector de coletor de arquivos).

```
$KAFKA_HOME/bin/connect-standalone.sh $KAFKA_HOME/config/connect-
standalone.properties $KAFKA_HOME/config/connect-file-sink.properties
```

Para obter um exemplo de uso do Kafka Connect, veja o script run-local-tests.sh na pasta de testes de integração no repositório do Github para o registro de esquemas do AWS Glue.

Migração de um registro de esquemas de terceiros para o registro de esquemas do AWS Glue

A migração de um registro de esquemas de terceiros para o registro de esquemas do AWS Glue tem uma dependência do registro de esquemas de terceiros existente e atual. Se houver registros em um tópico do Apache Kafka que tenham sido enviados usando um registro de esquemas de terceiros, os consumidores precisarão do registro de esquemas de terceiros para desserializar esses registros. O `AWSKafkaAvroDeserializer` fornece a capacidade de especificar uma classe secundária de desserialização que aponta para o desserializador de terceiros e é usada para desserializar esses registros.

Há dois critérios para desativação de um esquema de terceiros. Primeiro, a desativação pode ocorrer somente depois que os registros nos tópicos do Apache Kafka usando o registro de esquemas de terceiros não são mais exigidos por e para qualquer consumidor. Em segundo lugar, a desativação pode ocorrer com o vencimento dos tópicos do Apache Kafka, dependendo do período de retenção especificado para esses tópicos. Observe que, se você tiver tópicos com retenção infinita, ainda poderá migrar para o registro de esquemas do AWS Glue, mas não poderá desativar o registro de esquemas de terceiros. Como solução alternativa, você pode usar uma aplicação ou Mirror Maker 2 para ler o tópico atual e produzir para um novo tópico com o registro de esquemas do AWS Glue.

Migração de um registro de esquemas de terceiros para o registro de esquemas do AWS Glue:

1. Crie um registro no registro de esquemas do AWS Glue ou use o registro padrão.
2. Interrompa o consumidor. Modifique-o para incluir o registro de esquemas do AWS Glue como o principal desserializador e o registro de esquemas de terceiros como o secundário.
   - Defina as propriedades do consumidor. Neste exemplo, o secondary_deserializer é definido como um desserializador diferente. O comportamento é o seguinte: o consumidor recupera registros do Amazon MSK e primeiro tenta usar o `AWSKafkaAvroDeserializer`. Se não for possível ler o byte mágico que contém o ID do esquema Avro para o esquema do registro do esquemas do AWS Glue, o `AWSKafkaAvroDeserializer` tenta usar a classe do desserializador fornecida no secondary_deserializer. As propriedades específicas do desserializador secundário também precisam ser fornecidas nas propriedades do consumidor, como schema_registry_url_config e specific_avro_reader_config, como mostrado abaixo.

```
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 o consumidor.
4. Interrompa o produtor e aponte o produtor para o registro de esquemas do AWS Glue.
   a. Defina as propriedades do produtor. Neste exemplo, o produtor usará as versões de esquema de registro padrão e 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) Mova manualmente esquemas e versões de esquema existentes do registro de esquemas de terceiros atual para o registro de esquemas do AWS Glue, seja para o registro padrão no registro de esquemas do AWS Glue ou para um registro não padrão específico no registro de esquemas do AWS Glue. Isso pode ser feito exportando esquemas dos registros de esquemas de terceiros no formato JSON e criando novos esquemas no registro de esquemas do AWS Glue usando o AWS Management Console ou a AWS CLI.

   Esta etapa pode ser importante se você precisar habilitar verificações de compatibilidade com versões de esquema anteriores para versões de esquema recém-criadas usando a AWS CLI e o AWS Management Console. Ou quando os produtores enviam mensagens com um novo esquema com registro automático de versões de esquema ativado.
6. Inicie o produtor.

Conectar a dados

Uma AWS Glue conexão é um objeto do Catálogo de Dados que armazena credenciais de login, cadeias de caracteres de URI, informações de nuvem privada virtual (VPC) e muito mais para um armazenamento de dados específico. AWS Glue rastreadores, trabalhos e endpoints de desenvolvimento usam conexões para acessar determinados tipos de armazenamentos de dados. Você pode usar conexões para fontes e destinos, além de reutilizar a mesma conexão em vários trabalhos de crawler ou de extração, transformação e carregamento (ETL).

AWS Glue suporta os seguintes tipos de conexão:

- Amazon DocumentDB
- Amazon OpenSearch Service, para uso com o AWS Glue Spark.
- Amazon Redshift
- Azure Cosmos, para uso do Azure Cosmos DB para NoSQL com trabalhos de ETL AWS Glue
- Azure SQL, para uso com AWS Glue o Spark.
- Google BigQuery, para uso com o AWS Glue Spark.
- JDBC
- Kafka
- MongoDB
- MongoDB Atlas
- Salesforce
- SAP HANA, para uso com o AWS Glue Spark.
- Floco de neve, para uso com AWS Glue o Spark.
- Teradata Vantage, quando usado AWS Glue para o Spark.
- Vertica, para uso com o AWS Glue Spark.
- Várias ofertas do Amazon Relational Database Service (Amazon RDS).
- Rede (designa uma conexão para uma fonte de dados que esteja dentro de um ambiente da Amazon Virtual Private Cloud [Amazon VPC])
- Aurora (compatível se o driver JDBC nativo estiver sendo usado. Nem todos os atributos do driver podem ser aproveitados)

Com o AWS Glue Studio, você também pode criar uma conexão para um conector. Um conector é um pacote de código opcional que ajuda a acessar armazenamentos de dados no AWS Glue Studio. Para obter mais informações, consulte Uso de conectores e conexões com o AWS Glue Studio

Para obter informações sobre como se conectar a bancos de dados locais, consulte Como acessar e analisar armazenamentos de dados locais usando AWS Glue o site do AWS Big Data Blog.

Esta seção inclui os tópicos a seguir para ajudar você a usar conexões do AWS Glue :

- Propriedades da conexão do AWS Glue
- Armazenamento de credenciais de conexão no AWS Secrets Manager
- Adicionar uma conexão do AWS Glue
- Testar uma conexão do AWS Glue
- Configurar chamadas da AWS para passar por sua VPC
- Conectar-se a um armazenamento de dados JDBC em uma VPC
- Usar uma conexão do MongoDB ou do MongoDB Atlas
- Crawling em armazenamento de dados do Amazon S3 usando um endpoint da VPC
- Solução de problemas de conexão no AWS Glue
- Tutorial: Usar o AWS Glue Connector for Elasticsearch

Propriedades da conexão do AWS Glue

Este tópico inclui informações sobre propriedades para AWS Glue conexões.

Tópicos

- Requisitos de propriedades de conexão
- Propriedades da conexão JDBC do AWS Glue
- Propriedades de conexão do MongoDB e do MongoDB Atlas no AWS Glue
- Propriedades de conexão do Salesforce
- Conexão do Snowflake
- Conexão do Vertica
- Conexão do SAP HANA
- Conexão do Azure SQL
- Conexão do Teradata Vantage

- OpenSearch Conexão de serviço
- Conexão do Azure Cosmos
- Propriedades de conexão SSL do AWS Glue
- Propriedades de conexão do Apache Kafka para autenticação do cliente
- BigQuery Conexão do Google
- Conexão do Vertica

Requisitos de propriedades de conexão

Quando você define uma conexão no console do AWS Glue, é necessário fornecer valores para as seguintes propriedades:

Connection name (Nome da conexão)

Insira um nome exclusivo para a conexão.

Connection type

Escolha JDBC ou um dos tipos de conexão específicos.

Para obter detalhes sobre o tipo de conexão JDBC, consulte the section called “Propriedades da conexão JDBC”

Escolha Network (Rede) para conectar a uma origem dos dados dentro de um ambiente da Amazon Virtual Private Cloud (Amazon VPC).

Dependendo do tipo escolhido, o console do AWS Glue exibe outros campos obrigatórios. Por exemplo, se você escolher Amazon RDS, deverá escolher o mecanismo de banco de dados.

Exigir conexão SSL

Ao selecionar essa opção, o AWS Glue deve verificar se a conexão com o datastore está conectada por meio de uma Secure Sockets Layer (SSL) confiável.

Para obter mais informações, incluindo opções adicionais que estão disponíveis quando ao escolher essa opção, consulte the section called “Propriedades de conexão SSL”.

Selecione o cluster do MSK (somente Amazon Managed Streaming for Apache Kafka [MSK])

Especifica um cluster MSK de outra AWS conta.

URLs do servidor de bootstrap do Kafka (somente Kafka)

Especifica uma lista separada por vírgulas de URLs de servidor de bootstrap. Inclua o número da porta. Por exemplo: 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

Propriedades da conexão JDBC do AWS Glue

O AWS Glue pode se conectar aos seguintes armazenamentos de dados por meio de uma conexão JDBC:

- Amazon Redshift
- Amazon Aurora
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- Floco de neve, ao usar AWS Glue rastreadores.
- Aurora (compatível se o driver JDBC nativo estiver sendo usado. Nem todos os atributos do driver podem ser aproveitados)
- Amazon RDS for MariaDB

Important

Atualmente, um trabalho de ETL só pode usar uma conexão JDBC dentro de uma sub-rede. Se você tiver vários armazenamentos de dados em um trabalho, eles deverão estar na mesma sub-rede ou poder ser acessados na sub-rede.
Se você optar por trazer suas próprias versões do driver JDBC para os crawlers do AWS Glue, eles consumirão recursos em trabalhos do AWS Glue e do Amazon S3 para garantir que os drivers fornecidos sejam executados em seu ambiente. O uso adicional de recursos será refletido em sua conta. Além disso, fornecer seu próprio driver JDBC não significa que o crawler seja capaz de aproveitar todos os recursos do driver. Os drivers estão limitados às propriedades descritas em Definir conexões no catálogo de dados.

Veja a seguir as propriedades adicionais para o tipo de conexão JDBC.

JDBC URL

Insira a URL do datastore JDBC. Para a maioria dos mecanismos de banco de dados, este campo estará no seguinte formato. Nesse formato, substitua *protocol* (protocolo), *host*, *port* (porta) e *db_name* (nome do bd) por suas próprias informações.

```
jdbc:protocol://host:port/db_name
```

Dependendo do mecanismo de banco de dados, pode ser necessário um formato diferente de URL JDBC. Esse formato pode ter um uso um pouco diferente do ponto e vírgula (:) e da barra (/) ou conter palavras-chave diferentes para especificar bancos de dados.

Para que o JDBC se conecte ao datastore, é necessário ter um db_name no datastore. O db_name é usado para estabelecer uma conexão de rede com `username` e `password` fornecidos. Quando conectado, o AWS Glue pode acessar outros bancos de dados no datastore para executar um crawler ou um trabalho de ETL.

Os seguintes exemplos de URL JDBC mostram a sintaxe de vários mecanismos de banco de dados.

- Para se conectar a um datastore do cluster do Amazon Redshift com um banco de dados `dev`:

  ```
  jdbc:redshift://xxx.us-east-1.redshift.amazonaws.com:8192/dev
  ```

- Para se conectar a um datastore do Amazon RDS for MySQL com um banco de dados `employee`:

  ```
  jdbc:mysql://xxx-cluster.cluster-xxx.us-east-1.rds.amazonaws.com:3306/employee
  ```

- Para se conectar a um datastore do Amazon RDS for PostgreSQL com um banco de dados `employee`:

  ```
  jdbc:postgresql://xxx-cluster.cluster-xxx.us-east-1.rds.amazonaws.com:5432/employee
  ```

- Para se conectar a um datastore do Amazon RDS for Oracle com um nome de serviço `employee`:

  ```
  jdbc:oracle:thin://@xxx-cluster.cluster-xxx.us-east-1.rds.amazonaws.com:1521/employee
  ```

A sintaxe do Amazon RDS for Oracle pode seguir os seguintes padrões. Nesses padrões, substitua *host*, *port* (porta), *service_name* (nome do serviço) e *SID* por suas próprias informações.

- `jdbc:oracle:thin://@host:port/service_name`
- `jdbc:oracle:thin://@host:port:SID`

- Para se conectar a um datastore do Amazon RDS for Microsoft SQL Server com um banco de dados `employee`:

  ```
  jdbc:sqlserver://xxx-cluster.cluster-xxx.us-
  east-1.rds.amazonaws.com:1433;databaseName=employee
  ```

  A sintaxe do Amazon RDS for SQL Server pode seguir os seguintes padrões. Nesses padrões, substitua *server_name*, *port* e *db_name* com suas próprias informações.

  - `jdbc:sqlserver://server_name:port;database=db_name`
  - `jdbc:sqlserver://server_name:port;databaseName=db_name`

- Para se conectar a uma Amazon Aurora PostgreSQL instância do `employee` banco de dados, especifique o endpoint da instância do banco de dados, a porta e o nome do banco de dados:

  ```
  jdbc:postgresql://employee_instance_1.xxxxxxxxxxxx.us-
  east-2.rds.amazonaws.com:5432/employee
  ```

- Para se conectar a um armazenamento de Amazon RDS for MariaDB dados com um `employee` banco de dados, especifique o endpoint da instância do banco de dados, a porta e o nome do banco de dados:

  ```
  jdbc:mysql://xxx-cluster.cluster-xxx.aws-
  region.rds.amazonaws.com:3306/employee
  ```

- > **Warning**
  >
  > As conexões JDBC do Snowflake são suportadas somente por rastreadores. AWS Glue Ao usar o conector Snowflake em AWS Glue trabalhos, use o tipo de conexão Snowflake.

Para se conectar a uma instância do banco de dados `sample` do Snowflake, especifique o endpoint para a instância do Snowflake, o usuário, o nome do banco de dados e o nome do perfil. Você pode, opcionalmente, adicionar o parâmetro `warehouse`.

```
jdbc:snowflake://account_name.snowflakecomputing.com/?
user=user_name&db=sample&role=role_name&warehouse=warehouse_name
```

> **Important**
>
> Para conexões do Snowflake por JDBC, a ordem dos parâmetros na URL é imposta e deve ser ordenada como `user`, db `role_name` e `warehouse`.

- Para se conectar a uma instância do `sample` banco de dados do Snowflake com link AWS privado, especifique a URL JDBC do snowflake da seguinte forma:

```
jdbc:snowflake://account_name.region.privatelink.snowflakecomputing.com/?
user=user_name&db=sample&role=role_name&warehouse=warehouse_name
```

Nome de usuário

> **Note**
>
> Recomendamos que você use um AWS segredo para armazenar as credenciais de conexão em vez de fornecer seu nome de usuário e senha diretamente. Para ter mais informações, consulte Armazenamento de credenciais de conexão no AWS Secrets Manager.

Forneça um nome de usuário que tenha permissão para acessar o datastore JDBC.

Senha

Insira a senha para o nome de usuário que tem permissão de acesso ao datastore JDBC.

Porta

Insira a porta usada na URL do JDBC para se conectar a uma instância do Amazon RDS para Oracle. Esse campo é exibido somente quando a opção Require SSL connection (Exigir conexão SSL) é selecionada para uma instância do Amazon RDS Oracle.

VPC

Escolha o nome da nuvem privada virtual (VPC) que contém seu datastore. O console do AWS Glue lista todas as VPCs da região atual.

> **Important**
>
> Ao trabalhar em uma conexão JDBC hospedada fora do AWS, como com dados do Snowflake, sua VPC deve ter um gateway NAT que divida o tráfego em sub-redes públicas e privadas. A sub-rede pública é usada para conexão com a fonte externa e a sub-rede interna é usada para processamento por. AWS Glue Para obter informações sobre como configurar a Amazon VPC para conexões externas, leia Estabelecer conexão com a Internet ou a outras redes usando dispositivos NAT e Configurar um Amazon VPC para conexões JDBC aos armazenamentos de dados do Amazon RDS desde o AWS Glue.

Sub-rede

Escolha a sub-rede dentro da VPC que contém seu datastore. O console do AWS Glue lista todas as sub-redes do datastore na sua VPC.

Grupos de segurança

Escolha os grupos de segurança associados ao seu datastore. O AWS Glue requer um ou mais grupos de segurança com uma regra de origem de entrada que permita a conexão do AWS Glue. O console do AWS Glue lista todos os grupos de segurança que recebem acesso de entrada à sua VPC. O AWS Glue associa esses grupos de segurança à interface de rede elástica anexada à sua sub-rede VPC.

Nome da classe do driver JDBC - opcional

Forneça o nome da classe do driver 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

- SQL Server — com.microsoft.sqlserver.jdbc.sql ServerDriver

Caminho do driver JDBC do S3: optional

Forneça a localização do Amazon S3 para o driver JDBC personalizado. Esse é um caminho absoluto para um arquivo.jar. Se quiser fornecer seus próprios drivers JDBC para se conectar às fontes de dados dos bancos de dados compatíveis com crawlers, você pode especificar valores para parâmetros `customJdbcDriverS3Path` e `customJdbcDriverClassName`.
O uso de um driver JDBC fornecido por um cliente é limitado às Requisitos de propriedades de conexão necessárias.

Propriedades de conexão do MongoDB e do MongoDB Atlas no AWS Glue

As propriedades adicionais para o tipo de conexão do MongoDB ou do MongoDB Atlas são as que se seguem.

URL do MongoDB

Insira a URL do datastore MongoDB ou MongoDB Atlas:

- Para o MongoDB: mongodb://host:port/database. O host pode ser um nome de host, um endereço IP ou um soquete de domínio do UNIX. Se a string de conexão não especificar uma porta, a porta padrão do MongoDB, 27017, será usada.
- Para o MongoDB Atlas: mongodb+srv://server.example.com/database. O host pode ser um nome de host que corresponde a um registro SRV do DNS. O formato SRV não requer porta e usará a porta 27017, padrão do MongoDB.

Nome de usuário

> **Note**
>
> Recomendamos que você use um AWS segredo para armazenar as credenciais de conexão em vez de fornecer seu nome de usuário e senha diretamente. Para ter mais informações, consulte Armazenamento de credenciais de conexão no AWS Secrets Manager.

Forneça um nome de usuário que tenha permissão para acessar o datastore JDBC.

Senha

Insira a senha para o nome de usuário que tem permissão de acesso ao datastore MongoDB ou MongoDB Atlas.

Propriedades de conexão do Salesforce

Veja a seguir propriedades adicionais para o tipo de conexão do Salesforce.

- `ENTITY_NAME`(String) - (Obrigatório) Usado para leitura/gravação. O nome do seu objeto no Salesforce.
- `API_VERSION`(String) - (Obrigatório) Usado para leitura/gravação. Versão da API Rest do Salesforce que você deseja usar.
- `SELECTED_FIELDS`(Lista<String>) - Padrão: vazio (SELECIONE *). Usado para leitura. Colunas que você deseja selecionar para o objeto.
- `FILTER_PREDICATE`(String) - Padrão: vazio. Usado para leitura. Ele deve estar no formato Spark SQL.
- `QUERY`(String) - Padrão: vazio. Usado para leitura. Consulta completa do Spark SQL.
- `PARTITION_FIELD`(String) - Usado para leitura. Campo a ser usado para particionar a consulta.
- `LOWER_BOUND`(String) - Usado para leitura. Um valor limite inferior inclusivo do campo de partição escolhido.
- `UPPER_BOUND`(String) - Usado para leitura. Um valor limite superior exclusivo do campo de partição escolhido.
- `NUM_PARTITIONS`(Inteiro) - Padrão: 1. Usado para leitura. Número de partições para leitura.
- `IMPORT_DELETED_RECORDS`(String) - Padrão: FALSE. Usado para leitura. Para obter os registros excluídos durante a consulta.
- `WRITE_OPERATION`(String) - Padrão: INSERT. Usado para escrever. O valor deve ser INSERT, UPDATE, UPSERT, DELETE.
- `ID_FIELD_NAMES`(String) - Padrão: null. Usado somente para UPSERT.

Conexão do Snowflake

As propriedades a seguir são usadas para configurar uma conexão Snowflake usada em trabalhos de AWS Glue ETL. Ao fazer crawling no Snowflake, use uma conexão JDBC.

URL do Snowflake

A URL do endpoint do Snowflake. Para obter mais informações sobre as URLs de endpoint do Snowflake, consulte Connecting to Your Accounts na documentação do Snowflake.

AWS Segredo

O nome secreto de um segredo em AWS Secrets Manager. AWS Glue se conectará ao Snowflake usando as `sfPassword` chaves `sfUser` e do seu segredo.

Perfil do Snowflake (opcional)

Uma função de segurança do Snowflake AWS Glue será usada ao se conectar.

Use as propriedades a seguir ao configurar uma conexão com um endpoint do Snowflake hospedado na Amazon VPC usando AWS PrivateLink.

VPC

Escolha o nome da nuvem privada virtual (VPC) que contém seu datastore. O console do AWS Glue lista todas as VPCs da região atual.

Sub-rede

Escolha a sub-rede dentro da VPC que contém seu datastore. O console do AWS Glue lista todas as sub-redes do datastore na sua VPC.

Grupos de segurança

Escolha os grupos de segurança associados ao seu datastore. O AWS Glue requer um ou mais grupos de segurança com uma regra de origem de entrada que permita a conexão do AWS Glue. O console do AWS Glue lista todos os grupos de segurança que recebem acesso de entrada à sua VPC. O AWS Glue associa esses grupos de segurança à interface de rede elástica anexada à sua sub-rede VPC.

Conexão do Vertica

Use as propriedades a seguir para configurar uma conexão Vertica para trabalhos de AWS Glue ETL.

Host do Vertica

O nome do host da sua instalação do Vertica.

Porta do Vertica

A porta pela qual sua instalação do Vertica está disponível.

AWS Segredo

O nome secreto de um segredo em AWS Secrets Manager. AWS Glue se conectará à Vertica usando as chaves do seu segredo.

Use as propriedades a seguir ao configurar uma conexão com um endpoint do Vertica hospedado na Amazon VPC.

VPC

Escolha o nome da nuvem privada virtual (VPC) que contém seu datastore. O console do AWS Glue lista todas as VPCs da região atual.

Sub-rede

Escolha a sub-rede dentro da VPC que contém seu datastore. O console do AWS Glue lista todas as sub-redes do datastore na sua VPC.

Grupos de segurança

Escolha os grupos de segurança associados ao seu datastore. O AWS Glue requer um ou mais grupos de segurança com uma regra de origem de entrada que permita a conexão do AWS Glue. O console do AWS Glue lista todos os grupos de segurança que recebem acesso de entrada à sua VPC. O AWS Glue associa esses grupos de segurança à interface de rede elástica anexada à sua sub-rede VPC.

Conexão do SAP HANA

Use as propriedades a seguir para configurar uma conexão SAP HANA para trabalhos de AWS Glue ETL.

URL DO SAP HANA

Um URL JDBC do SAP.

Os URLs JDBC do SAP HANA estão no formato `jdbc:sap://saphanaHostname:saphanaPort/?databaseName=saphanaDBname,ParameterNa`

AWS Glue requer os seguintes parâmetros de URL do JDBC:

- `databaseName`: um banco de dados padrão no SAP HANA ao qual se conectar.

AWS Segredo

O nome secreto de um segredo em AWS Secrets Manager. AWS Glue se conectará ao SAP HANA usando as chaves do seu segredo.

Use as seguintes propriedades ao configurar uma conexão com um endpoint do SAP HANA hospedado na Amazon VPC:

VPC

Escolha o nome da nuvem privada virtual (VPC) que contém seu datastore. O console do AWS Glue lista todas as VPCs da região atual.

Sub-rede

Escolha a sub-rede dentro da VPC que contém seu datastore. O console do AWS Glue lista todas as sub-redes do datastore na sua VPC.

Grupos de segurança

Escolha os grupos de segurança associados ao seu datastore. O AWS Glue requer um ou mais grupos de segurança com uma regra de origem de entrada que permita a conexão do AWS Glue. O console do AWS Glue lista todos os grupos de segurança que recebem acesso de entrada à sua VPC. O AWS Glue associa esses grupos de segurança à interface de rede elástica anexada à sua sub-rede VPC.

Conexão do Azure SQL

Use as propriedades a seguir para configurar uma conexão SQL do Azure para trabalhos de AWS Glue ETL.

URL do Azure SQL

O URL JDBC de um endpoint do Azure SQL.

Essa lista deve estar no seguinte formato: `jdbc:sqlserver://databaseServerName:databasePort;databaseName=azuresqlDBname;`.

AWS Glue requer as seguintes propriedades de URL:

- `databaseName`: um banco de dados padrão no Azure SQL ao qual se conectar.

Para obter mais informações sobre URLs de JDBC para instâncias gerenciadas Azure SQL, consulte a Documentação da Microsoft.

AWS Segredo

O nome secreto de um segredo em AWS Secrets Manager. AWS Glue se conectará ao Azure SQL usando as chaves do seu segredo.

Conexão do Teradata Vantage

Use as propriedades a seguir para configurar uma conexão Teradata Vantage para AWS Glue trabalhos de ETL.

URL do Teradata

Para se conectar a uma instância do Teradata, especifique o nome do host da instância do banco de dados e os parâmetros relevantes do Teradata:

```
jdbc:teradata://teradataHostname/ParameterName=ParameterValue,ParameterName=Pa
```

AWS Glue suporta os seguintes parâmetros de URL do JDBC:

- `DATABASE_NAME`: um banco de dados padrão no Teradata ao qual se conectar.
- `DBS_PORT`: especifica a porta do Teradata, caso não seja a padrão.

AWS Segredo

O nome secreto de um segredo em AWS Secrets Manager. AWS Glue se conectará ao Teradata Vantage usando as chaves do seu segredo.

Use as seguintes propriedades ao configurar uma conexão com um endpoint do Teradata Vantage hospedado na Amazon VPC:

VPC

Escolha o nome da nuvem privada virtual (VPC) que contém seu datastore. O console do AWS Glue lista todas as VPCs da região atual.

Sub-rede

Escolha a sub-rede dentro da VPC que contém seu datastore. O console do AWS Glue lista todas as sub-redes do datastore na sua VPC.

Grupos de segurança

Escolha os grupos de segurança associados ao seu datastore. O AWS Glue requer um ou mais grupos de segurança com uma regra de origem de entrada que permita a conexão do AWS Glue. O console do AWS Glue lista todos os grupos de segurança que recebem acesso de entrada à sua VPC. O AWS Glue associa esses grupos de segurança à interface de rede elástica anexada à sua sub-rede VPC.

OpenSearch Conexão de serviço

Use as propriedades a seguir para configurar uma conexão OpenSearch de serviço para trabalhos de AWS Glue ETL.

Endpoint de domínio

Um endpoint OpenSearch de domínio do Amazon Service terá o seguinte formato padrão, https://search - *`DomainName`* -. *`unstructuredIdContent`* *`região`* .es.amazonaws.com. Para obter mais informações sobre como identificar o endpoint do seu domínio, consulte Criação e gerenciamento de domínios do Amazon OpenSearch Service na documentação do Amazon OpenSearch Service.

Porta

A porta aberta no endpoint.

AWS Segredo

O nome secreto de um segredo em AWS Secrets Manager. AWS Glue se conectará ao OpenSearch Serviço usando as chaves do seu segredo.

Use as seguintes propriedades ao configurar uma conexão com um endpoint de OpenSearch serviço hospedado na Amazon VPC:

VPC

Escolha o nome da nuvem privada virtual (VPC) que contém seu datastore. O console do AWS Glue lista todas as VPCs da região atual.

Sub-rede

Escolha a sub-rede dentro da VPC que contém seu datastore. O console do AWS Glue lista todas as sub-redes do datastore na sua VPC.

Grupos de segurança

Escolha os grupos de segurança associados ao seu datastore. O AWS Glue requer um ou mais grupos de segurança com uma regra de origem de entrada que permita a conexão do AWS Glue. O console do AWS Glue lista todos os grupos de segurança que recebem acesso de entrada à sua VPC. O AWS Glue associa esses grupos de segurança à interface de rede elástica anexada à sua sub-rede VPC.

Conexão do Azure Cosmos

Use as propriedades a seguir para configurar uma conexão do Azure Cosmos para trabalhos de AWS Glue ETL.

URI do endpoint da conta do Azure Cosmos DB

O endpoint usado para se conectar ao Azure Cosmos. Para obter mais informações, consulte a Documentação do Azure.

AWS Segredo

O nome secreto de um segredo em AWS Secrets Manager. AWS Glue se conectará ao Azure Cosmos usando as chaves do seu segredo.

Propriedades de conexão SSL do AWS Glue

Veja a seguir detalhes sobre a propriedade Require SSL connection (Exigir conexão SSL).

Se você não precisar de uma conexão SSL, o AWS Glue ignorará falhas ao usar SSL para criptografar uma conexão com o datastore. Consulte a documentação do datastore para obter instruções de configuração. Quando você selecionar essa opção, a execução de trabalho, o crawler ou as declarações de ETL em um endpoint falharão quando o AWS Glue não conseguir estabelecer conexão.

> **Note**
>
> O Snowflake é compatível com uma conexão SSL por padrão, portanto, essa propriedade não se aplica ao Snowflake.

Essa opção é validada no lado do cliente do AWS Glue. Para conexões JDBC, o AWS Glue apenas se conecta via SSL com validação de certificado e nome de host. O suporte à conexão SSL está disponível para:

- Oracle Database
- Microsoft SQL Server
- PostgreSQL
- Amazon Redshift
- MySQL (apenas instâncias do Amazon RDS)
- Amazon Aurora MySQL (apenas instâncias do Amazon RDS)
- Amazon Aurora PostgreSQL (Somente instâncias do Amazon RDS)
- Kafka, que inclui Amazon Managed Streaming for Apache Kafka
- MongoDB

> **Note**
>
> Para ativar um datastore do Amazon RDS Oracle para usar Require SSL connection (Exigir conexão SSL), é necessário criar e anexar um grupo de opções à instância do Oracle.
>
> 1. Faça login AWS Management Console e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
> 2. Adicione um Option group (Grupo de opções) à instância do Amazon RDS Oracle. Para obter mais informações sobre como adicionar um grupo de opções no console do Amazon RDS, consulte Criar um grupo de opções
> 3. Adicione uma Opção ao grupo de opções para SSL. A porta que você especifica para o SSL é posteriormente usada quando você cria um URL de conexão JDBC do AWS Glue para a instância do Amazon RDS Oracle. Para obter mais informações sobre como adicionar uma opção no console do Amazon RDS, consulte Adicionar uma opção a um grupo de opções no Manual do usuário do Amazon RDS. Para obter mais informações sobre a opção Oracle SSL, consulte Oracle SSL no Manual do usuário do Amazon RDS.
> 4. No console do AWS Glue, crie uma conexão com a instância do Amazon RDS Oracle. Na definição da conexão, selecione Require SSL connection (Exigir conexão SSL). Quando solicitado, insira a porta usada na opção Oracle SSL do Amazon RDS.

As propriedades opcionais adicionais a seguir estarão disponíveis quando Require SSL connection (Exigir conexão SSL) for selecionado para uma conexão:

Certificado JDBC personalizado no S3

Se tiver um certificado que está sendo usado para comunicação SSL com os bancos de dados on-premises ou na nuvem, você poderá usar esse certificado para conexões SSL com destinos ou fontes de dados do AWS Glue. Insira um local do Amazon Simple Storage Service (Amazon S3) que contenha um certificado raiz personalizado. O AWS Glue usa esse certificado para estabelecer uma conexão SSL com o banco de dados. O AWS Glue lida apenas com certificados X.509. O certificado deve ter codificação DER e ser fornecido no formato PEM de codificação base64.

Se esse campo for deixado em branco, o certificado padrão será usado.

String de certificado JDBC personalizado

Insira informações de certificado específicas do banco de dados JDBC. Essa string é usada para correspondência de domínio ou para correspondência de nome distinto (DN). Para o Oracle Database, essa string é mapeada para o parâmetro `SSL_SERVER_CERT_DN` na seção de segurança do arquivo `tnsnames.ora`. Para o Microsoft SQL Server, essa string é usada como `hostNameInCertificate`.

Veja a seguir um exemplo para o parâmetro `SSL_SERVER_CERT_DN` do Oracle Database.

```
cn=sales,cn=OracleContext,dc=us,dc=example,dc=com
```

Localização do certificado CA privado do Kafka

Se você tiver um certificado que está usando atualmente para comunicação SSL com seu datastore do Kafka, poderá usar esse certificado com a sua conexão do AWS Glue. Essa opção é necessária para armazenamentos de dados Kafka e opcional para armazenamentos de Amazon Managed Streaming for Apache Kafka dados. Insira um local do Amazon Simple Storage Service (Amazon S3) que contenha um certificado raiz personalizado. O AWS Glue usa esse certificado para estabelecer uma conexão SSL com o datastore do Kafka. O AWS Glue lida apenas com certificados X.509. O certificado deve ter codificação DER e ser fornecido no formato PEM de codificação base64.

Ignorar validação de certificado

Marque a caixa de seleção Skip certificate validation (Ignorar validação do certificado) para ignorar a validação do certificado personalizado pelo AWS Glue. Se você optar por validar, o

AWS Glue validará o algoritmo de assinatura e o algoritmo de chave pública de assunto para o certificado. Se o certificado falhar na validação, qualquer trabalho de ETL ou crawler que usar a conexão falhará.

Os únicos algoritmos de assinatura permitidos são SHA256withRSA, SHA384withRSA ou SHA512withRSA. Para o algoritmo de chave pública do assunto, o comprimento da chave deve ser pelo menos 2048.

Localização do repositório de chaves do cliente Kafka

A localização no Amazon S3 do arquivo de repositório de chaves para autenticação do lado do cliente Kafka. O caminho deve estar no formato s3://bucket/prefix/filename.jks. Ele deve terminar com o nome do arquivo e a extensão .jks.

Senha do repositório de chaves do cliente Kafka (opcional)

A senha para acessar o repositório de chaves fornecido.

Senha da chave do cliente Kafka (opcional)

Um repositório de chaves pode consistir em várias chaves. Essa é a senha para acessar a chave do cliente a ser usada com a chave do lado do servidor Kafka.

Propriedades de conexão do Apache Kafka para autenticação do cliente

O AWS Glue é compatível com a estrutura Simple Authentication and Security Layer (SASL) para autenticação quando você cria uma conexão do Apache Kafka. A estrutura SASL é compatível com vários mecanismos de autenticação e o AWS Glue oferece os protocolos SCRAM (nome de usuário e senha) e GSSAPI (protocolo Kerberos) e PLAIN.

Use AWS Glue Studio para configurar um dos seguintes métodos de autenticação do cliente. Para obter mais informações, consulte Criação de conexões para conectores no guia do AWS Glue Studio usuário.

- Nenhum - Sem autenticação. Isso é útil se criar uma conexão para fins de teste.
- SASL/SCRAM-SHA-512 - Escolher esse método de autenticação permitirá que você especifique credenciais de autenticação. Existem duas opções disponíveis:
  - Use o AWS Secrets Manager (recomendado) - se você selecionar essa opção, poderá armazenar seu nome de usuário e senha no AWS Secrets Manager e permitir o AWS Glue acesso a eles quando necessário. Especifique o segredo que armazena as credenciais de

autenticação SSL ou SASL. Para ter mais informações, consulte Armazenamento de credenciais de conexão no AWS Secrets Manager.

- Forneça um nome de usuário e senha diretamente.

- SASL/GSSAPI (Kerberos) - se você selecionar essa opção, poderá selecionar o local do arquivo keytab, arquivo krb5.conf e inserir o nome principal do Kerberos e o nome do serviço Kerberos. Os locais do arquivo keytab e do arquivo krb5.conf devem estar em um local do Amazon S3. Como o MSK ainda não oferece suporte a SASL/GSSAPI, essa opção está disponível apenas para clusters Apache Kafka gerenciados pelo cliente. Para obter mais informações, consulte Documentação do MIT Kerberos: Keytab.
- SASL/PLAIN: escolha esse método de autenticação para especificar as credenciais de autenticação. Existem duas opções disponíveis:
  - Use o AWS Secrets Manager (recomendado) - se você selecionar essa opção, poderá armazenar suas credenciais no AWS Secrets Manager e permitir o AWS Glue acesso às informações quando necessário. Especifique o segredo que armazena as credenciais de autenticação SSL ou SASL.
  - Forneça o nome de usuário e a senha diretamente.
- Autenticação de cliente SSL - se você selecionar essa opção, poderá selecionar o local do keystore do cliente Kafka navegando no Amazon S3. Opcionalmente, você pode inserir a senha do keystore do cliente Kafka e a senha da chave do cliente Kafka.

BigQuery Conexão do Google

As propriedades a seguir são usadas para configurar uma BigQuery conexão com o Google usada em trabalhos de AWS Glue ETL. Para ter mais informações, consulte the section called “Conexões BigQuery”.

AWS Segredo

O nome secreto de um segredo em AWS Secrets Manager. AWS Glue As tarefas de ETL se conectarão ao Google BigQuery usando a `credentials` chave do seu segredo.

Conexão do Vertica

As propriedades a seguir são usadas para configurar uma conexão Vertica usada em trabalhos de AWS Glue ETL. Para ter mais informações, consulte the section called “Conexões do Vertica”.

Armazenamento de credenciais de conexão no AWS Secrets Manager

Recomendamos usar o AWS Secrets Manager para fornecer credenciais de conexão para seu armazenamento de dados. Usar o Secrets Manager dessa maneira permite que o AWS Glue acesse seu segredo em runtime para trabalhos de ETL e execuções de crawler, além de ajudar a manter suas credenciais seguras.

Pré-requisitos

Para usar o Secrets Manager com o AWS Glue, você deve conceder permissão para que seu perfil do IAM para o AWS Glue recupere valores de segredos. A política gerenciada pela AWS `AWSGlueServiceRole` não inclui as permissões AWS Secrets Manager. Para obter exemplos de políticas do IAM, consulte Exemplo: permissão para recuperar valores de segredos no Guia do usuário do AWS Secrets Manager.

Dependendo da sua configuração de rede, talvez também seja necessário criar um endpoint da VPC para estabelecer uma conexão privada entre sua VPC e o Secrets Manager. Para obter mais informações, consulte Usar um endpoint da VPC no AWS Secrets Manager.

Para criar um segredo para o AWS Glue

1. Siga as instruções em Criar e gerenciar segredos no Guia do usuário do AWS Secrets Manager. O exemplo de JSON a seguir mostra como especificar suas credenciais na guia Plaintext (Texto não criptografado) ao criar um segredo para o AWS Glue.

   ```
   {
     "username": "EXAMPLE-USERNAME",
     "password": "EXAMPLE-PASSWORD"
   }
   ```

2. Associe um segredo a uma conexão usando a interface do AWS Glue Studio. Para obter as instruções detalhadas, consulte Creating connections for connectors no AWS Glue Studio User Guide.

Adicionar uma conexão do AWS Glue

Você pode se conectar a fontes de dados no Spark AWS Glue de forma programática. Para mais informações, consulte Tipos e opções de conexão para ETL no AWS Glue para Spark.

É possível usar também o console do AWS Glue para adicionar, editar, excluir e testar conexões. Para obter informações sobre as conexões do AWS Glue, consulte Conectar a dados.

Como adicionar uma conexão do AWS Glue

1. Faça login no AWS Management Console e abra o AWS Glue console em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (catálogo de dados), selecione Connections (Conexões).
3. Selecione Add connection (Adicionar conexão) e conclua o assistente, inserindo as propriedades da conexão conforme descrito em the section called “Propriedades da conexão do AWS Glue”.

Conectando-se ao Amazon Redshift em AWS Glue Studio

> **Note**
>
> Você pode usar AWS Glue o Spark para ler e gravar tabelas em Amazon Redshift bancos de dados externos ao. AWS Glue Studio Para configurar Amazon Redshift com AWS Glue trabalhos de forma programática, consulteConexões do Redshift.

AWS Gluefornece suporte integrado para Amazon Redshift. AWS Glue Studiofornece uma interface visual para se conectar Amazon Redshift, criar trabalhos de integração de dados e executá-los no tempo de execução do AWS Glue Studio Spark sem servidor.

Tópicos

- Criando uma Amazon Redshift conexão
- Criar um nó de origem do Amazon Redshift
- Criação de um nó de Amazon Redshift destino
- Opções avançadas

Criando uma Amazon Redshift conexão

Permissões necessárias

São necessárias permissões adicionais para usar Amazon Redshift clusters e ambientes Amazon Redshift sem servidor. Para obter mais informações sobre como adicionar permissões a trabalhos de ETL, consulte Review IAM permissions needed for ETL jobs.

- desvio para o vermelho: DescribeClusters
- redshift sem servidor: ListWorkgroups
- redshift sem servidor: ListNamespaces

Visão geral

Ao adicionar uma Amazon Redshift conexão, você pode escolher uma Amazon Redshift conexão existente ou criar uma nova conexão ao adicionar um nó Fonte de dados - Redshift. AWS Glue Studio

AWS Glueoferece suporte a Amazon Redshift clusters e Amazon Redshift ambientes sem servidor. Quando você cria uma conexão, os ambientes Amazon Redshift sem servidor exibem o rótulo sem servidor ao lado da opção de conexão.

Para obter mais informações sobre como criar uma Amazon Redshift conexão, consulte Como mover dados de e para Amazon Redshift.

Criar um nó de origem do Amazon Redshift

Permissões necessárias

Trabalhos do AWS Glue Studio usando fontes de dados do Amazon Redshift exigem permissões adicionais. Para obter mais informações sobre como adicionar permissões a trabalhos de ETL, consulte Review IAM permissions needed for ETL jobs.

As seguintes permissões são necessárias para usar uma conexão com Amazon Redshift.

- redshift-data:ListSchemas
- redshift-data:ListTables
- redshift-data:DescribeTable
- redshift-data:ExecuteStatement

- redshift-data:DescribeStatement
- redshift-data:GetStatementResult

Adicionar de uma fonte de dados do Amazon Redshift

Para adicionar um nó fonte de dados - Amazon Redshift:

1. Escolha o tipo de acesso ao Amazon Redshift:

   - Conexão direta de dados (recomendada): escolha essa opção se quiser acessar seus dados do Amazon Redshift diretamente. Essa é a opção recomendada e também a padrão.
   - Data Catalog tables: escolha essa opção se você tiver tabelas do catálogo de dados que deseja usar.

2. Se você escolher Conexão de dados direta, escolha a conexão para sua fonte de dados do Amazon Redshift. Isso pressupõe que a conexão já exista e que você possa selecionar entre as conexões existentes. Se precisar criar uma conexão, escolha Criar conexão com o Redshift. Para obter mais informações, consulte Visão geral do uso de conectores e conexões.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades. As informações sobre a conexão estão visíveis, incluindo URL, grupos de segurança, sub-rede, zona de disponibilidade, descrição e timestamps de criação (UTC) e última atualização (UTC).

3. Escolha uma opção de origem do Amazon Redshift:

   - Escolher uma única tabela: essa é a tabela que contém os dados que você deseja acessar de uma única tabela do Amazon Redshift.
   - Inserir consulta personalizada: permite que você acesse um conjunto de dados de várias tabelas do Amazon Redshift com base na sua consulta personalizada.

4. Se você escolheu uma única tabela, escolha o esquema do Amazon Redshift. A lista de esquemas disponíveis para escolha é determinada pela tabela selecionada.

   Ou escolha Inserir consulta personalizada. Escolha essa opção para acessar um conjunto de dados personalizado de várias tabelas do Amazon Redshift. Ao escolher essa opção, insira a consulta do Amazon Redshift.

   Ao se conectar a um ambiente Amazon Redshift sem servidor, adicione a seguinte permissão à consulta personalizada:

```
GRANT SELECT ON ALL TABLES IN <schema> TO PUBLIC
```

Você pode escolher Inferir esquema para ler o esquema com base na consulta que você inseriu. Você também pode escolher Abrir editor de consultas do Redshift para inserir uma consulta do Amazon Redshift. Para obter mais informações, acesse Consultar um banco de dados usando o editor de consultas.

5. Em Desempenho e segurança, escolha o diretório de teste do Amazon S3 e o perfil do IAM.
   - Diretório de teste do Amazon S3: escolha o local do Amazon S3 para armazenar temporariamente os dados do teste.
   - Perfil do IAM: escolha o perfil do IAM que pode gravar no local do Amazon S3 que você selecionou.
6. Em Parâmetros personalizados do Redshift - opcional, insira o parâmetro e o valor.

Criação de um nó de Amazon Redshift destino

Permissões necessárias

AWS Glue Studio trabalhos que usam o destino de Amazon Redshift dados exigem permissões adicionais. Para obter mais informações sobre como adicionar permissões a trabalhos de ETL, consulte Review IAM permissions needed for ETL jobs.

As permissões a seguir são necessárias para usar uma Amazon Redshift conexão.

- dados do redshift: ListSchemas
- dados do redshift: ListTables

Adicionando um nó de Amazon Redshift destino

Para criar um nó de Amazon Redshift destino:

1. Escolha uma Amazon Redshift tabela existente como destino ou insira um novo nome de tabela.
2. Ao usar o nó de destino Destino de dados: Redshift, você pode escolher entre as seguintes opções:

- APPEND: se uma tabela já existir, despejar todos os novos dados na tabela como uma inserção. Se a tabela não existir, criar e inserir todos os novos dados.

  Além disso, marque a caixa se quiser atualizar (UPSERT) os registros existentes na tabela de destino. A tabela deve existir primeiro, caso contrário, a operação falhará.

- MERGE: o AWS Glue atualizará ou anexará dados à sua tabela de destino com base nas condições que você especificar.

  > **Note**
  >
  > Para usar a ação de mesclagem emAWS Glue, você deve habilitar a funcionalidade de Amazon Redshift mesclagem. Para obter instruções sobre como habilitar a mesclagem para sua Amazon Redshift instância, consulte MERGE (pré-visualização).

  Escolha as opções:

  - Escolher chaves e ações simples: escolha as colunas a serem usadas como chaves de correspondência entre os dados de origem e seu conjunto de dados de destino.

    Especifique as seguintes opções quando correspondidas:

    - Atualizar o registro em seu conjunto de dados de destino com os dados da fonte.
    - Excluir o registro em seu conjunto de dados de destino.

    Especifique as seguintes opções quando não correspondidas:

    - Inserir os dados de origem como uma nova linha em seu conjunto de dados de destino.
    - Não executar nenhuma ação.

  - Inserir declaração MERGE personalizada: você pode então escolher Validar declaração de Merge para verificar se a declaração é válida ou inválida.

- TRUNCATE: se uma tabela já existir, truncar os dados da tabela limpando primeiro o conteúdo da tabela de destino. Se o truncamento for bem-sucedido, inserir todos os dados. Se a tabela não existir, criar a tabela e inserir todos os novos dados. Se o truncamento não tiver sucesso, a operação falhará.
- DROP: se uma tabela já existir, excluir os metadados e os dados da tabela. Se a exclusão for bem-sucedida, inserir todos os dados. Se a tabela não existir, criar a tabela e inserir todos os novos dados. Se o drop não tiver sucesso, a operação falhará.

- CREATE: Criar uma nova tabela com o nome padrão. Se o nome da tabela já existir, criar uma nova tabela com um aposto do nome de `job_datetime` para fins de exclusividade. Isso inserirá todos os dados na nova tabela. Se a tabela existir, o nome final da tabela terá o aposto anexado. Se a tabela não existir, uma tabela será criada. Em ambos os casos, uma nova tabela será criada.

Opções avançadas

Consulte Using the Amazon Redshift Spark connector AWS Glue.

Conectar ao Azure Cosmos DB no AWS Glue Studio

O AWS Glue oferece suporte integrado ao Azure Cosmos DB. O AWS Glue Studio fornece uma interface visual para conectar ao Azure Cosmos DB para NoSQL, criar trabalhos de integração de dados e executá-los no runtime Spark do AWS Glue Studio sem servidor.

Tópicos

Criar uma conexão do Azure Cosmos DB

Pré-requisitos:

- No Azure, será necessário identificar ou gerar uma chave do Azure Cosmos DB para uso pelo AWS Glue, `cosmosKey`. Para obter mais informações, consulte Acesso seguro a dados no Azure Cosmos DB na documentação do Azure.

Para configurar uma conexão com o Azure Cosmos DB:

1. No AWS Secrets Manager, crie um segredo usando sua chave do Azure Cosmos DB. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *`SecretName`*, para a próxima etapa.

- Ao selecionar Pares chave/valor, crie um par para a chave `spark.cosmos.accountKey` com o valor *`cosmosKey`*.

2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *`connectionName`*, para uso futuro em AWS Glue.

   - Ao selecionar um Tipo de conexão, selecione Azure Cosmos DB.
   - Ao selecionar um Segredo da AWS, forneça o *`secretName`*.

Criar um nó de origem do Azure Cosmos DB

Pré-requisitos necessários

- Uma conexão AWS Glue Azure Cosmos DB configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called "Criar uma conexão do Azure Cosmos DB".
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Um contêiner do Azure Cosmos DB para NoSQL do qual você gostaria de ler. Você precisará de informações de identificação para o contêiner.

  Um contêiner do Azure Cosmos para NoSQL é identificado por seu banco de dados e contêiner. É necessário fornecer os nomes do banco de dados, *`cosmosDBName`* e do contêiner, *`cosmosContainerName`*, ao se conectar à API do Azure Cosmos para NoSQL.

Adicionar uma fonte de dados do Azure Cosmos DB

Para adicionar um nó de Fonte de dados – Azure Cosmos DB:

1. Escolha a conexão para sua fonte de dados do Azure Cosmos DB. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão do Azure Cosmos DB. Para mais informações, consulte a seção the section called "Criar uma conexão do Azure Cosmos DB" anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.

2. Escolha o Nome do banco de dados do Cosmos DB: forneça o nome do banco de dados do qual você deseja ler, *cosmosDBName*.
3. Escolha Contêiner do Azure Cosmos DB: forneça o nome do contêiner do qual você deseja ler, *cosmosContainerName*.
4. Opcionalmente, escolha Consulta personalizada do Azure Cosmos DB: forneça uma consulta SQL SELECT para recuperar informações específicas do Azure Cosmos DB.
5. Em Propriedades personalizadas do Azure Cosmos, insira parâmetros e valores conforme necessário.

Criar um nó de destino do Azure Cosmos DB

Pré-requisitos necessários

- Uma conexão AWS Glue Azure Cosmos DB configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called “Criar uma conexão do Azure Cosmos DB”.
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Uma tabela do Azure Cosmos DB em que você deseja escrever. Você precisará de informações de identificação para o contêiner. É necessário criar o contêiner antes de chamar o método de conexão.

  Um contêiner do Azure Cosmos para NoSQL é identificado por seu banco de dados e contêiner. É necessário fornecer os nomes do banco de dados, *cosmosDBName* e do contêiner, *cosmosContainerName*, ao se conectar à API do Azure Cosmos para NoSQL.

Adicionar um destino de dados do Azure Cosmos DB

Para adicionar um nó de Destino de dados – Azure Cosmos DB:

1. Escolha a conexão para sua fonte de dados do Azure Cosmos DB. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão do Azure Cosmos DB. Para mais informações, consulte a seção the section called “Criar uma conexão do Azure Cosmos DB” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.

2. Escolha o Nome do banco de dados do Cosmos DB: forneça o nome do banco de dados do qual você deseja ler, *cosmosDBName*.
3. Escolha Contêiner do Azure Cosmos DB: forneça o nome do contêiner do qual você deseja ler, *cosmosContainerName*.
4. Em Propriedades personalizadas do Azure Cosmos, insira parâmetros e valores conforme necessário.

Opções avançadas

É possível fornecer opções avançadas ao criar um nó do Azure Cosmos DB. Essas opções são as mesmas disponíveis ao programar AWS Glue para scripts do Spark.

Consulte the section called “Conexões do Azure Cosmos DB”.

Conectar ao Azure SQL no AWS Glue Studio

O AWS Glue oferece suporte integrado ao Azure SQL. O AWS Glue Studio fornece uma interface visual para conectar ao Azure SQL, criar trabalhos de integração de dados e executá-los no runtime Spark do AWS Glue Studio sem servidor.

Tópicos

Criar uma conexão do Azure SQL

Para se conectar ao Azure SQL via AWS Glue, será necessário criar e armazenar sua credenciais do Azure SQL em um segredo do AWS Secrets Manager e, em seguida, associar esse segredo a uma conexão do Azure SQL AWS Glue.

Para configurar uma conexão com o Azure SQL:

1. No AWS Secrets Manager, crie um segredo usando suas credenciais do Azure SQL. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager

segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *SecretName*, para a próxima etapa.

- Ao selecionar Pares chave/valor, crie um par para a chave `user` com o valor *azuresqlUsername*.
- Ao selecionar Pares chave/valor, crie um par para a chave `password` com o valor *azuresqlPassword*.

2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *connectionName*, para uso futuro em AWS Glue.

   - Ao selecionar um Tipo de conexão, selecione Azure SQL.
   - Ao fornecer o URL do Azure SQL, forneça um URL de endpoint do JDBC.

     Essa lista deve estar no seguinte formato: `jdbc:sqlserver://`*databaseServerName*`:`*databasePort*`;databaseName=`*azuresqlDBnam*

     O AWS Glue requer as seguintes propriedades de URL:

     - `databaseName`: um banco de dados padrão no Azure SQL ao qual se conectar.

     Para obter mais informações sobre URLs de JDBC para instâncias gerenciadas Azure SQL, consulte a Documentação da Microsoft.

   - Ao selecionar um Segredo da AWS, forneça o *secretName*.

Criar um nó de origem do Azure SQL

Pré-requisitos necessários

- Uma conexão AWS Glue Azure SQL configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called "Criar uma conexão do Azure SQL".
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Uma tabela do Azure SQL da qual você deseja ler, *tableName*.

  Uma tabela do Azure SQL é identificada por seu banco de dados, esquema e nome da tabela. É necessário fornecer o nome do banco de dados e o nome da tabela ao se conectar ao Azure SQL. Você também deverá fornecer o esquema se ele não for o padrão, "public". O banco de dados

é fornecido por meio de uma propriedade de URL em *connectionName*, esquema e nome da tabela via `dbtable`.

Adicionar uma fonte de dados do Azure SQL

Para adicionar um nó de Fonte de dados – Azure SQL:

1. Escolha a conexão para sua fonte de dados do Azure SQL. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão do Azure SQL. Para mais informações, consulte a seção the section called “Criar uma conexão do Azure SQL” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Escolha uma opção de Fonte do Azure SQL:
   - Escolher uma única tabela: acesse todos os dados de uma única tabela.
   - Inserir consulta personalizada: acesse um conjunto de dados de várias tabelas com base em sua consulta personalizada.
3. Se você escolheu uma única tabela, insira *tableName*.

   Se você escolheu Inserir consulta personalizada, insira uma consulta TransactSQL SELECT.
4. Em Propriedades personalizadas do Azure SQL, insira parâmetros e valores conforme necessário.

Criar um nó de destino do Azure SQL

Pré-requisitos necessários

- Uma conexão AWS Glue Azure SQL configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called “Criar uma conexão do Azure SQL”.
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Uma tabela do Azure SQL em que você deseja escrever, *tableName*.

  Uma tabela do Azure SQL é identificada por seu banco de dados, esquema e nome da tabela. É necessário fornecer o nome do banco de dados e o nome da tabela ao se conectar ao Azure SQL. Você também deverá fornecer o esquema se ele não for o padrão, "public". O banco de dados

é fornecido por meio de uma propriedade de URL em *connectionName*, esquema e nome da tabela via dbtable.

Adicionar um destino de dados do Azure SQL

Para adicionar um nó de Destino de dados – Azure SQL:

1. Escolha a conexão para sua fonte de dados do Azure SQL. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão do Azure SQL. Para mais informações, consulte a seção the section called “Criar uma conexão do Azure SQL” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Configure o nome da tabela fornecendo o *tableName*.
3. Em Propriedades personalizadas do Azure SQL, insira parâmetros e valores conforme necessário.

Opções avançadas

É possível fornecer opções avançadas ao criar um nó do Azure SQL. Essas opções são as mesmas disponíveis ao programar AWS Glue para scripts do Spark.

Consulte the section called “Conexões do Azure SQL”.

Conectando-se ao Google BigQuery em AWS Glue Studio

> **Note**
>
> Você pode usar o AWS Glue Spark para ler e gravar em tabelas no Google BigQuery nas versões AWS Glue 4.0 e posteriores. Para configurar o Google BigQuery com AWS Glue trabalhos de forma programática, consulte Conexões BigQuery.

AWS Glue Studiofornece uma interface visual para se conectar BigQuery, criar trabalhos de integração de dados e executá-los no tempo de execução do AWS Glue Studio Spark sem servidor.

Tópicos

Criar uma conexão BigQuery

Para se conectar ao Google BigQuery a partir do AWS Glue, você precisará criar e armazenar suas credenciais do Google Cloud Platform em um segredo e, em seguida, associar esse AWS Secrets Manager segredo a uma conexão do Google BigQuery AWS Glue.

Para configurar uma conexão com o BigQuery:

1. No Google Cloud Platform, crie e identifique recursos relevantes:
   - Crie ou identifique um projeto do GCP contendo tabelas do BigQuery às quais você gostaria de se conectar.
   - Ative a API BigQuery. Para obter mais informações, consulte Use the BigQuery Storage Read API to read na tabela.
2. No Google Cloud Platform, crie e exporte as credenciais da conta de serviço:

   Você pode usar o assistente de credenciais do BigQuery para acelerar essa etapa: criar credenciais.

   Para criar uma conta de serviço no GCP, siga o tutorial disponível em Criar contas de serviço.
   - Ao selecionar o projeto, selecione o projeto que contém sua tabela do BigQuery.
   - Ao selecionar perfis do IAM do GCP para sua conta de serviço, adicione ou crie um papel que conceda permissões apropriadas para executar jobs do BigQuery para ler, gravar ou criar tabelas do BigQuery.

   Para criar credenciais para a sua conta de serviço, siga o tutorial disponível em Criar uma chave da conta de serviço.
   - Ao selecionar o tipo de chave, selecione JSON.

Agora você deve ter baixado um arquivo JSON com credenciais para sua conta de serviço. A aparência deve ser semelhante à seguinte:

```
{
  "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. base64 codifica seu arquivo de credenciais baixado. Em uma AWS CloudShell sessão ou similar, você pode fazer isso na linha de comando executando `cat` *`credentialsFile.json`* `| base64 -w 0`. Mantenha a saída desse comando, *`credentialString`*.
4. Nos AWS Secrets Manager, crie um segredo usando suas credenciais do Google Cloud Platform. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *`SecretName`*, para a próxima etapa.
   - *Ao selecionar* ***pares de chave/valor****, crie um par para a chave credentials com o valor credentialString.*
5. No catálogo de dados do AWS Glue, crie uma conexão seguindo as etapas em https://docs.aws.amazon.com/glue/latest/dg/console-connections.html. Depois de criar a conexão, guarde o nome da conexão, *`connectionName`*, para a próxima etapa.
   - Ao selecionar um tipo de conexão, selecione Google BigQuery.
   - Ao selecionar um Segredo da AWS, forneça o *`secretName`*.
6. Conceda ao perfil do IAM associada ao seu trabalho do AWS Glue permissão para ler *`secretName`*.
7. Na configuração do trabalho do AWS Glue, forneça *`connectionName`* como uma conexão de rede adicional.

Criar um nó de origem do BigQuery

Pré-requisitos necessários

- Uma conexão de catálogo de dados do AWS Glue do tipo Snowflake
- Um AWS Secrets Manager segredo para suas credenciais do Google BigQuery, usado pela conexão.
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- O nome e o conjunto de dados da tabela e do projeto do Google Cloud correspondente no qual você deseja gravar.

Adicionar uma fonte de dados do BigQuery

Para adicionar um destino de dados — nó do BigQuery:

1. Escolha a conexão para seu destino de dados do BigQuery. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão com o Redshift. Para obter mais informações, consulte Visão geral do uso de conectores e conexões.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Identifique quais dados do BigQuery você gostaria de ler e escolha uma opção de origem do BigQuery

   - Escolha uma única tabela — permite extrair todos os dados de uma tabela.
   - Insira uma consulta personalizada — permite que você personalize quais dados são recuperados fornecendo uma consulta.
3. Descreva os dados que você gostaria de ler

   (Obrigatório) defina Projeto principal como o projeto que contém sua tabela ou um projeto principal de cobrança, se relevante.

   Se você escolheu uma única tabela, defina Tabela com o nome de uma tabela do Google BigQuery no seguinte formato: `[dataset].[table]`

   Se você escolheu uma consulta, forneça-a ao Query. Em sua consulta, consulte as tabelas com seu nome de tabela totalmente qualificado, no formato:`[project].[dataset].[tableName]`.

4. Forneça propriedades do BigQuery

   Se você escolheu uma única tabela, não é necessário fornecer propriedades adicionais.

   Se você escolher uma consulta, deverá fornecer as seguintes propriedades personalizadas do Google BigQuery:

   - Defina `viewsEnabled` como verdadeiro.
   - Defina `materializationDataset` como um conjunto de dados. O principal do GCP autenticado pelas credenciais fornecidas pela AWS Glue conexão precisa ser capaz de criar tabelas nesse conjunto de dados.

Criar um nó de destino do BigQuery

Pré-requisitos necessários

- Uma conexão de catálogo de dados do AWS Glue do tipo Snowflake
- Um AWS Secrets Manager segredo para suas credenciais do Google BigQuery, usado pela conexão.
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- O nome e o conjunto de dados da tabela e do projeto do Google Cloud correspondente no qual você deseja gravar.

Adicionar um destino de dados do BigQuery

Para adicionar um destino de dados — nó do BigQuery:

1. Escolha a conexão para seu destino de dados do BigQuery. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão com o Redshift. Para obter mais informações, consulte Visão geral do uso de conectores e conexões.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Identifique em qual tabela do BigQuery você gostaria de gravar e escolha um método de gravação.

   - Direto — grava diretamente no BigQuery usando a API BigQuery Storage Write.
   - Indireto — grava no Google Cloud Storage e depois copia no BigQuery.

Se você quiser escrever indiretamente, forneça um local de destino do GCS com um bucket temporário do GCS. Você precisará fornecer configurações adicionais em sua AWS Glue conexão. Para obter mais informações, consulte Usar a gravação indireta com o Google BigQuery.

3. Descreva os dados que você gostaria de ler

   (Obrigatório) defina Projeto principal como o projeto que contém sua tabela ou um projeto principal de cobrança, se relevante.

   Se você escolheu uma única tabela, defina Tabela com o nome de uma tabela do Google BigQuery no seguinte formato: `[dataset].[table]`

Opções avançadas

Você pode fornecer opções avançadas ao criar um BigQuery nó. Essas opções são as mesmas disponíveis ao programar AWS Glue para scripts do Spark.

Consulte BigQuery a referência da opção de conexão no guia do AWS Glue desenvolvedor.

Conectar ao MongoDB no AWS Glue Studio

O AWS Glue oferece suporte integrado ao Mongo DB. O AWS Glue Studio fornece uma interface visual para conectar ao Mongo DB, criar trabalhos de integração de dados e executá-los no runtime Spark do AWS Glue Studio sem servidor.

Tópicos

- Criar uma conexão do MongoDB
- Criar um nó de origem do MongoDB
- Criar um nó de destino do MongoDB
- Opções avançadas

Criar uma conexão do MongoDB

Pré-requisitos:

- Se a sua instância do MongoDB estiver em uma Amazon VPC, configure a Amazon VPC para permitir que seu trabalho do AWS Glue se comunique com a instância do MongoDB sem que o tráfego passe pela Internet pública.

  Na Amazon VPC, identifique ou crie uma VPC, uma Sub-rede e um Grupo de segurança que o AWS Glue usará durante a execução do trabalho. Além disso, você precisa garantir que a Amazon VPC esteja configurada para permitir o tráfego de rede entre sua instância do MongoDB e esse local. Com base no layout da rede, isso pode exigir alterações em regras do grupo de segurança, ACLs de rede, gateways de NAT e conexões de emparelhamento.

Para configurar uma conexão com o MongoDB:

1. Opcionalmente, no AWS Secrets Manager, crie um segredo usando suas credenciais do MongoDB. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *SecretName*, para a próxima etapa.

   - Ao selecionar Pares chave/valor, crie um par para a chave `username` com o valor *mongodbUser*.

     Ao selecionar Pares chave/valor, crie um par para a chave `password` com o valor *mongodbPass*.

2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *connectionName*, para uso futuro em AWS Glue.

   - Ao selecionar um Tipo de conexão, selecione MongoDB ou MongoDB Atlas.
   - Ao selecionar o URL do MongoDB ou o URL do MongoDB Atlas, forneça o nome de host da sua instância do MongoDB.

     Um URL do MongoDB é fornecido no formato `mongodb://`*mongoHost*`:`*mongoPort*`/`*mongoDBname*.

     Um URL do MongoDB Atlas é fornecido no formato `mongodb+srv://`*mongoHost*`:`*mongoPort*`/`*mongoDBname*.

     Fornecer o banco de dados padrão *mongoDBname* para a conexão é opcional.

- Se você optar por criar um segredo do Secrets Manager, escolha o Tipo de credencial do AWS Secrets Manager.

  Em seguida, em Segredo do AWS, forneça o *secretName*.

- *Se você optar por fornecer **Nome de usuário e senha**, forneça também o mongodbUser e o mongodbPass*.

3. Nas seguintes situações, configurações adicionais podem ser necessárias:

   - Para instâncias do MongoDB hospedadas na AWS em uma Amazon VPC
     - Será necessário fornecer informações de conexão da Amazon VPC à conexão do AWS Glue que define suas credenciais de segurança do MongoDB. Ao criar ou atualizar sua conexão, defina VPC, Sub-rede e Grupos de segurança em Opções de rede.

Depois de criar uma conexão AWS Glue MongoDB, siga estas etapas antes de executar seu trabalho do AWS Glue:

- Ao trabalhar com trabalhos do AWS Glue no editor visual, é necessário fornecer informações de conexão da Amazon VPC para que seu trabalho se conecte ao MongoDB. Identifique um local adequado na Amazon VPC e forneça-o à sua conexão AWS Glue MongoDB.
- Se você optar por criar um segredo do Secrets Manager, conceda ao perfil do IAM associado ao seu trabalho do AWS Glue permissão para ler *secretName*.

Criar um nó de origem do MongoDB

Pré-requisitos necessários

- Uma conexão AWS Glue MongoDB, conforme descrito na seção anterior, the section called “Criar uma conexão do MongoDB”.
- Se você optar por criar um segredo do Secrets Manager, permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Uma coleção do MongoDB da qual você gostaria de ler. Você precisará de informações de identificação para a coleção.

  Uma coleção do MongoDB é identificada por um nome de banco de dados e um nome de coleção, *mongodbName* e *mongodbCollection*.

Adicionar uma fonte de dados do MongoDB

Para adicionar um nó de Fonte de dados – MongoDB:

1. Escolha a conexão para sua fonte de dados do MongoDB. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão com o MongoDB. Para mais informações, consulte a seção the section called “Criar uma conexão do MongoDB” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Escolha um Banco de dados. Insira o *`mongodbName`*.
3. Escolha uma Coleção. Insira *`mongodbCollection`*.
4. Escolha seu Particionador, o Tamanho da partição (MB) e a Chave da partição. Para obter mais informações sobre parâmetros de partições, consulte the section called ““connectionType”: “mongodb” como fonte”.
5. Em Propriedades personalizadas do MongoDB, insira parâmetros e valores conforme necessário.

Criar um nó de destino do MongoDB

Pré-requisitos necessários

- Uma conexão AWS Glue MongoDB configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called “Criar uma conexão do MongoDB”.
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Uma tabela do MongoDB na qual você gostaria de escrever, *`tableName`*.

Adicionar um destino de dados do MongoDB

Para adicionar um nó de Destino de dados – MongoDB:

1. Escolha a conexão para sua fonte de dados do MongoDB. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão com o MongoDB. Para mais informações, consulte a seção the section called “Criar uma conexão do MongoDB” anterior.

Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.

2. Escolha um Banco de dados. Insira o *mongodbName*.
3. Escolha uma Coleção. Insira *mongodbCollection*.
4. Escolha seu Particionador, o Tamanho da partição (MB) e a Chave da partição. Para obter mais informações sobre parâmetros de partições, consulte the section called ““connectionType”: “mongodb” como fonte”.
5. Escolha Repetir operações de escrever, se desejar.
6. Em Propriedades personalizadas do MongoDB, insira parâmetros e valores conforme necessário.

Opções avançadas

Você pode fornecer opções avançadas ao criar um nó do Mongo DB. Essas opções são as mesmas disponíveis ao programar AWS Glue para scripts do Spark.

Consulte the section called “Conexão do MongoDB”.

Conectar ao OpenSearch Service no AWS Glue Studio

O AWS Glue oferece suporte integrado ao Amazon OpenSearch Service. O AWS Glue Studio fornece uma interface visual para conectar ao Amazon OpenSearch Service, criar trabalhos de integração de dados e executá-los no runtime Spark do AWS Glue Studio sem servidor. Esse recurso não é compatível com o OpenSearch Service com tecnologia sem servidor.

Tópicos

Criar uma conexão com o OpenSearch Service

Pré-requisitos:

- Identifique o endpoint do domínio, *aosEndpoint* e porta, *aosPort* do qual gostaria de ler, ou crie o recurso seguindo as instruções na documentação do Amazon OpenSearch Service. Para obter mais informações sobre a criação de domínios, consulte Criar e gerenciar domínios do Amazon OpenSearch Service na documentação do Amazon OpenSearch Service.

  Um endpoint de domínio do Amazon OpenSearch Service terá o seguinte formato padrão, https://search-*domainName*-*unstructuredIdContent*.*region*.es.amazonaws.com. Para obter mais informações sobre a identificação do endpoint do domínio, consulte Criar e gerenciar domínios do Amazon OpenSearch Service na documentação do Amazon OpenSearch Service.

  Identifique ou gere credenciais de autenticação básica HTTP, *aosUser* e *aosPassword* para seu domínio.

Para configurar uma conexão com o OpenSearch Service:

1. No AWS Secrets Manager, crie um segredo usando suas credenciais do OpenSearch Service. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *SecretName*, para a próxima etapa.

   - Ao selecionar Pares chave/valor, crie um par para a chave `opensearch.net.http.auth.user` com o valor *aosUser*.
   - Ao selecionar Pares chave/valor, crie um par para a chave `opensearch.net.http.auth.pass` com o valor *aosPassword*.
2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *connectionName*, para uso futuro em AWS Glue.

   - Ao selecionar um Tipo de conexão, escolha OpenSearch Service.
   - Ao selecionar um endpoint de domínio, forneça *aosEndpoint*.
   - Ao selecionar uma porta, forneça *aosPort*.
   - Ao selecionar um Segredo da AWS, forneça o *secretName*.

Criar um nó de origem do OpenSearch Service

Pré-requisitos necessários

- Uma conexão do AWS Glue OpenSearch Service, configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called “Criar uma conexão com o OpenSearch Service”.
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Um índice do OpenSearch Service do qual você gostaria de ler, *aosIndex*.

Adicionar uma fonte de dados do OpenSearch Service

Para adicionar um nó de Fonte de dados – OpenSearch Service:

1. Escolha a conexão para sua fonte de dados do OpenSearch Service. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão com o OpenSearch Service. Para mais informações, consulte a seção the section called “Criar uma conexão com o OpenSearch Service” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Forneça o Índice que você gostaria de ler.
3. Opcionalmente, forneça Consulta, uma consulta do OpenSearch para fornecer resultados mais específicos. Para obter mais informações sobre como escrever consultas do OpenSearch, consulte the section called “Ler do OpenSearch Service”.
4. Em Propriedades personalizadas do OpenSearch Service, insira parâmetros e valores conforme o necessário.

Criar um nó de destino do OpenSearch Service

Pré-requisitos necessários

- Uma conexão do AWS Glue OpenSearch Service, configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called “Criar uma conexão com o OpenSearch Service”.
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Um índice do OpenSearch Service em que você gostaria de escrever, *aosIndex*.

Adicionar um destino de dados do OpenSearch Service

Para adicionar um nó de Destino de dados – OpenSearch Service:

1. Escolha a conexão para sua fonte de dados do OpenSearch Service. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão com o OpenSearch Service. Para mais informações, consulte a seção the section called “Criar uma conexão com o OpenSearch Service” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Forneça o Índice que você gostaria de ler.
3. Em Propriedades personalizadas do OpenSearch Service, insira parâmetros e valores conforme o necessário.

Opções avançadas

Você pode fornecer opções avançadas ao criar um nó do OpenSearch Service. Essas opções são as mesmas disponíveis ao programar AWS Glue para scripts do Spark.

Consulte the section called “Conexões do OpenSearch Service”.

Conectando-se ao Salesforce em AWS Glue Studio

A Salesforce fornece software de gerenciamento de relacionamento com o cliente (CRM) que ajuda você com vendas, atendimento ao cliente, comércio eletrônico e muito mais. Se você é usuário do Salesforce, pode se conectar AWS Glue à sua conta do Salesforce. Em seguida, você pode usar o Salesforce como fonte de dados ou destino em seus trabalhos de ETL. Execute essas tarefas para transferir dados entre o Salesforce e AWS os serviços ou outros aplicativos compatíveis.

Tópicos

- AWS Glue suporte para Salesforce
- Políticas contendo as operações de API para criar e usar conexões
- Configurando o Salesforce
- Configurando conexões do Salesforce
- Leitura de entidades da Salesforce
- Escrevendo para a Salesforce

- Opções de conexão do Salesforce
- Limitações do conector Salesforce
- Configurar o fluxo OAuth do portador JWT para Salesforce

AWS Glue suporte para Salesforce

AWS Glue oferece suporte ao Salesforce da seguinte forma:

Compatível como fonte?

Sim. Você pode usar trabalhos AWS Glue ETL para consultar dados do Salesforce.

Suportado como alvo?

Sim. Você pode usar o AWS Glue ETL para gravar registros no Salesforce.

Versões compatíveis da API Salesforce

As seguintes versões da API Salesforce são suportadas

- v58.0
- v59.0
- v60.0

Políticas contendo as operações de API para criar e usar conexões

O exemplo de política a seguir descreve as permissões necessárias AWS do IAM para criar e usar conexões. Se você estiver criando um novo perfil, crie uma política que contenha o seguinte:

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Você também pode usar as seguintes políticas do IAM para permitir o acesso:

- AWSGlueServiceRole— Concede acesso aos recursos que vários AWS Glue processos exigem para serem executados em seu nome. Esses recursos incluem AWS Glue Amazon S3, IAM, CloudWatch Logs e Amazon EC2. Se você seguir a convenção de nomenclatura dos recursos especificados nesta política, AWS Glue os processos terão as permissões necessárias. Esta política geralmente é anexada a funções especificadas durante a definição de crawlers, trabalhos e endpoints de desenvolvimento.
- AWSGlueConsoleFullAccess— Concede acesso total aos AWS Glue recursos quando uma identidade à qual a política está anexada usa o AWS Management Console. Se você seguir a convenção de nomenclatura para os recursos especificados nesta política, os usuários poderão acessar todos os recursos do console. Essa política geralmente é anexada aos usuários do AWS Glue console.

Configurando o Salesforce

Antes de usar AWS Glue para transferir dados para ou do Salesforce, você deve atender aos seguintes requisitos:

Requisitos mínimos

A seguir estão os requisitos mínimos:

- Você tem uma conta do Salesforce.
- Sua conta do Salesforce está habilitada para acesso à API. O acesso à API é ativado por padrão para as edições Enterprise, Unlimited, Developer e Performance.
- Sua conta do Salesforce permite que você instale aplicativos conectados. Se você não tiver acesso a essa funcionalidade, entre em contato com seu administrador do Salesforce. Para obter mais informações, consulte Aplicativos conectados na ajuda do Salesforce.

Se você atender a esses requisitos, estará pronto para se conectar AWS Glue à sua conta do Salesforce. AWS Glue lida com os requisitos restantes com o aplicativo conectado AWS gerenciado.

O aplicativo conectado AWS gerenciado para Salesforce

O aplicativo conectado AWS gerenciado ajuda você a criar conexões com o Salesforce em menos etapas. No Salesforce, um aplicativo conectado é uma estrutura que autoriza aplicativos externos, por exemplo AWS Glue, a acessar seus dados do Salesforce.

- Crie uma conexão com o Salesforce usando o AWS Glue console.
- Ao configurar a conexão, defina o tipo de concessão OAuth como Código de autorização.

Configurando conexões do Salesforce

Para configurar uma conexão com o Salesforce:

1. No AWS Secrets Manager, crie um segredo com os seguintes detalhes:
   a. Para o tipo de concessão JWT_TOKEN, o segredo deve conter a chave JWT_TOKEN com seu valor.
   b. Para o tipo de AuthorizationCode concessão: para um aplicativo conectado gerenciado pelo cliente, o segredo deve conter o aplicativo conectado Consumer Secret com `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET` como chave. Para um aplicativo conectado AWS gerenciado, um segredo vazio ou um segredo com algum valor temporário.
   c. Nota: Você deve criar um segredo por conexão em AWS Glue.
2. No Catálogo AWS Glue de Dados, crie uma conexão seguindo as etapas abaixo:
   a. Ao selecionar um tipo de conexão, selecione Salesforce.
   b. Forneça o INSTANCE_URL do Salesforce ao qual você deseja se conectar.
   c. Forneça o ambiente Salesforce.
   d. Selecione a função AWS do IAM que AWS Glue pode assumir e tem permissões para as seguintes ações:
   e. Selecione o tipo de concessão do OAuth2 que você deseja usar para as conexões. O tipo de concessão determina como AWS Glue se comunica com a Salesforce para solicitar acesso aos seus dados. Sua escolha afeta os requisitos que você deve atender antes de criar a conexão. Você pode escolher um desses tipos:
      - Tipo de concessão JWT_BEARER: Esse tipo de concessão funciona bem para cenários de automação, pois permite que um token Web JSON (JWT) seja criado antecipadamente com as permissões de um usuário específico na instância do Salesforce. O criador tem controle

sobre por quanto tempo o JWT é válido. AWS Glue é capaz de usar o JWT para obter um token de acesso que é usado para chamar as APIs do Salesforce.

Esse fluxo exige que o usuário tenha criado um aplicativo conectado em sua instância do Salesforce, o que permite a emissão de tokens de acesso baseados em JWT para os usuários.

Para obter informações sobre como criar um aplicativo conectado para o fluxo OAuth do portador JWT, consulte Fluxo do portador JWT do OAuth 2.0 para integração. server-to-server Para configurar o fluxo do portador JWT com o aplicativo conectado Salesforce, consulte. Configurar o fluxo OAuth do portador JWT para Salesforce

- Tipo de concessão AUTHORIZATION_CODE: Esse tipo de concessão é considerado um OAuth de “três pernas”, pois depende do redirecionamento dos usuários para o servidor de autorização de terceiros para autenticar o usuário. Ele é usado ao criar conexões por meio do AWS Glue console. O usuário que está criando uma conexão pode, por padrão, confiar em um aplicativo AWS Glue conectado (aplicativo cliente AWS Glue gerenciado), no qual não precisa fornecer nenhuma informação relacionada ao OAuth, exceto a URL da instância do Salesforce. O AWS Glue console redirecionará o usuário para o Salesforce, onde o usuário deve fazer login e permitir que as permissões solicitadas acessem sua instância AWS Glue do Salesforce.

  Os usuários ainda podem optar por criar seu próprio aplicativo conectado no Salesforce e fornecer seu próprio ID e segredo do cliente ao criar conexões por meio do AWS Glue console. Nesse cenário, eles ainda serão redirecionados ao Salesforce para fazer login e autorizar o acesso AWS Glue a seus recursos.

  Esse tipo de concessão resulta em um token de atualização e um token de acesso. O token de acesso tem vida curta e pode ser atualizado automaticamente sem a interação do usuário usando o token de atualização.

  Para obter informações sobre como criar um aplicativo conectado para o fluxo OAuth do Código de Autorização, consulte Configurar um aplicativo conectado para o fluxo de código de autorização e credenciais.

f. Selecione o `secretName` que você deseja usar para essa conexão AWS Glue para colocar os tokens.

g. Selecione as opções de rede se quiser usar sua rede. Conceda permissão de leitura ao papel do IAM associado ao seu AWS Glue trabalho`secretName`.

3. Conceda permissão de leitura ao papel do IAM associado ao seu AWS Glue trabalho`secretName`.
4. Em sua configuração de AWS Glue trabalho, forneça `connectionName` como uma conexão de rede adicional.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

Leitura de entidades da Salesforce

Pré-requisito

Um sObject do Salesforce que você gostaria de ler. Você precisará do nome do objeto, como `Account` ou `Case` ou`Opportunity`.

Exemplo:

```
salesforce_read = glueContext.create_dynamic_frame.from_options(
    connection_type="salesforce",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Account",
        "API_VERSION": "v60.0"
    }
```

Consultas de particionamento

Você pode fornecer as opções adicionais do SparkPARTITION_FIELD,LOWER_BOUND,UPPER_BOUND, e NUM_PARTITIONS se quiser utilizar a simultaneidade no Spark. Com esses parâmetros, a consulta original seria dividida em NUM_PARTITIONS várias subconsultas que podem ser executadas pelas tarefas do Spark simultaneamente.

- PARTITION_FIELD: o nome do campo a ser usado para particionar a consulta.
- LOWER_BOUND: um valor limite inferior inclusivo do campo de partição escolhido.

  Para o campo de timestamp, aceitamos o formato de timestamp do Spark usado nas consultas SQL do Spark.

  Exemplos 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: um valor limite superior exclusivo do campo de partição escolhido.
- NUM_PARTITIONS: o número de partições.

Exemplo:

```
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"
    }
```

Escrevendo para a Salesforce

Pré-requisitos

Um sObject do Salesforce para o qual você gostaria de escrever. Você precisará do nome do objeto, como `Account` ou `Case` ou`Opportunity`.

O conector Salesforce suporta quatro operações de gravação:

- INSERT
- UPSERT
- UPDATE
- DELETE

Ao usar a operação de `UPSERT` gravação, é `ID_FIELD_NAMES` necessário fornecer o campo de ID externo para os registros.

Exemplo

```
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"
    }
```

Opções de conexão do Salesforce

A seguir estão as opções de conexão para o Salesforce:

- `ENTITY_NAME`(String) - (Obrigatório) Usado para leitura/gravação. O nome do seu objeto no Salesforce.
- `API_VERSION`(String) - (Obrigatório) Usado para leitura/gravação. Versão da API Rest do Salesforce que você deseja usar.
- `SELECTED_FIELDS`(Lista<String>) - Padrão: vazio (SELECIONE *). Usado para leitura. Colunas que você deseja selecionar para o objeto.

- `FILTER_PREDICATE`(String) - Padrão: vazio. Usado para leitura. Ele deve estar no formato Spark SQL.
- `QUERY`(String) - Padrão: vazio. Usado para leitura. Consulta completa do Spark SQL.
- `PARTITION_FIELD`(String) - Usado para leitura. Campo a ser usado para particionar a consulta.
- `LOWER_BOUND`(String) - Usado para leitura. Um valor limite inferior inclusivo do campo de partição escolhido.
- `UPPER_BOUND`(String) - Usado para leitura. Um valor limite superior exclusivo do campo de partição escolhido.
- `NUM_PARTITIONS`(Inteiro) - Padrão: 1. Usado para leitura. Número de partições para leitura.
- `IMPORT_DELETED_RECORDS`(String) - Padrão: FALSE. Usado para leitura. Para obter os registros excluídos durante a consulta.
- `WRITE_OPERATION`(String) - Padrão: INSERT. Usado para escrever. O valor deve ser INSERT, UPDATE, UPSERT, DELETE.
- `ID_FIELD_NAMES`(String) - Padrão: null. Usado somente para UPSERT.

Limitações do conector Salesforce

A seguir estão as limitações do conector Salesforce:

- Só oferecemos suporte ao Spark SQL e o Salesforce SOQL não é suportado.
- Marcadores de trabalho não são compatíveis.

Configurar o fluxo OAuth do portador JWT para Salesforce

Consulte a documentação pública da Salesforce para habilitar a server-to-server integração com os Tokens Web JSON do OAuth 2.0.

Criação de um par cert/chave de arquivos PEM

Crie um par cert/chave de arquivos PEM

```
openssl req -newkey rsa:4096 -new -nodes -x509 -days 3650 -keyout key.pem -out cert.pem
```

Criação de um aplicativo conectado ao Salesforce com o JWT

1. Faça login no Salesforce, clique na engrenagem de configurações no canto superior direito e selecione Configuração.
2. À esquerda, navegue até o App Manager. (Ferramentas da plataforma > Aplicativos > Gerenciador de aplicativos)
3. Escolha Novo aplicativo de conexão.
4. Forneça um nome de aplicativo e deixe o resto ser preenchido automaticamente.
5. Marque a caixa Ativar configurações de OAuth.
6. Defina um URL de retorno de chamada. Ele não será usado para JWT, então você pode usar https://localhost.
7. Marque a caixa Usar assinaturas digitais.
8. Faça upload do arquivo cert.pem criado anteriormente.
9. Adicione as permissões necessárias:
   a. Gerencie os dados do usuário por meio de APIs (api).
   b. Acesse permissões personalizadas (custom_permissions).
   c. Acesse o serviço de URL de identidade (ID, perfil, e-mail, endereço, telefone).
   d. Acesse identificadores de usuário exclusivos (openid).
   e. Execute solicitações a qualquer momento (refresh_token, offline_access).
10. Marque a caixa para emitir tokens de acesso baseados em JSON Web Token (JWT) para usuários nomeados.
11. Escolha Salvar.
12. Escolha Continuar.
13. Escolha Gerenciar detalhes do consumidor.
14. Copie a chave do consumidor (ID do cliente).
15. Copie o segredo do consumidor (segredo do cliente).
16. Clique em Cancel.

Gerando um token web JSON (JWT)

1. Converta o par de chaves em pkcs12 (defina uma senha de exportação quando solicitado).

```
openssl pkcs12 -export -in cert.pem -inkey key.pem -name jwtcert > jwtcert.p12
```

2. Crie um repositório de chaves Java a partir do pkcs12 (defina uma senha do armazenamento de chaves de destino quando solicitado e forneça a senha de exportação anterior para a senha do armazenamento de chaves de origem).

```
keytool -importkeystore -srckeystore jwtcert.p12 -destkeystore keystore.jks -
srcstoretype pkcs12 -alias jwtcert
```

3. Confirme se keystore.jks inclui o alias jwtcert (insira a senha anterior do keystore de destino quando solicitado).

```
keytool -keystore keystore.jks -list
```

4. Use a classe Java JwtExample fornecida na documentação do Salesforce para gerar o token assinado.
   a. Edite os valores em ClaimArray conforme necessário:
      - ClaimArray [0] = id do cliente
      - claimArray [1] = ID de usuário do salesforce
      - claimArray [2] = URL de login do salesforce
      - ClaimArray [4] = data de expiração em millis desde a época. 3660624000000 é 2085-12-31.
   b. Substitua path/to/keystore pelo caminho correto para seu keystore.jks.
   c. Substitua keystorepassword pela senha do keystore de destino que você inseriu
   d. Substitua privatekeypassword pela senha do keystore de origem que você inseriu.
   e. Compile o código. O código depende do Codec Apache Commons para codificação base64.

```
javac -classpath ".:./commons-codec-1.16.1.jar" JWTExample.java
```

   f. Execute o código.

```
java -classpath ".:commons-codec-1.16.1.jar" JWTExample
```

5. Depois que o aplicativo conectado e o JWT são criados, o usuário ainda precisa ser autorizado para usar o aplicativo. Consulte a etapa 3 em https://mannharleen.github.io/2020-03-03-salesforce-jwt/ para ver duas abordagens.

Com as etapas acima concluídas, isso deve gerar um JSON Web Token (JWT) que pode ser usado para obter tokens de acesso do Salesforce.

Exemplo 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
```

Exemplos de resultado:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0Ij
```

Links úteis:

- 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();
    }
  }
}
```

Conectar ao SAP HANA no AWS Glue Studio

O AWS Glue oferece suporte integrado ao SAP HANA. O AWS Glue Studio fornece uma interface visual para conectar ao SAP HANA, criar trabalhos de integração de dados e executá-los no runtime Spark do AWS Glue Studio sem servidor.

Tópicos

- Criar uma conexão SAP HANA
- Criar um nó de origem do SAP HANA
- Criar um nó de destino do SAP HANA
- Opções avançadas

Criar uma conexão SAP HANA

Para se conectar ao SAP HANA via AWS Glue, será necessário criar e armazenar suas credenciais do SAP HANA em um segredo do AWS Secrets Manager e, em seguida, associar esse segredo a uma conexão ao AWS Glue do SAP HANA. Você precisará configurar a conectividade de rede entre seu serviço SAP HANA e o AWS Glue.

Pré-requisitos:

- Se o seu serviço SAP HANA estiver em uma Amazon VPC, configure a Amazon VPC para permitir que seu trabalho do AWS Glue se comunique com o serviço SAP HANA sem que o tráfego passe pela Internet pública.

  Na Amazon VPC, identifique ou crie uma VPC, uma Sub-rede e um Grupo de segurança que o AWS Glue usará durante a execução do trabalho. Além disso, você precisa garantir que a Amazon VPC esteja configurada para permitir o tráfego de rede entre seu endpoint SAP HANA e esse local. Seu trabalho precisará estabelecer uma conexão TCP com a porta JDBC do SAP HANA. Para obter mais informações sobre as portas do SAP HANA, consulte a Documentação do SAP HANA.

Com base no layout da rede, isso pode exigir alterações em regras do grupo de segurança, ACLs de rede, gateways de NAT e conexões de emparelhamento.

Para configurar uma conexão com o SAP HANA:

1. No AWS Secrets Manager, crie um segredo usando suas credenciais do SAP HANA. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *`SecretName`*, para a próxima etapa.

   - Ao selecionar Pares chave/valor, crie um par para a chave `user` com o valor *`saphanaUsername`*.
   - Ao selecionar Pares chave/valor, crie um par para a chave `password` com o valor *`saphanaPassword`*.

2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *`connectionName`*, para uso futuro em AWS Glue.

   - Ao selecionar um Tipo de conexão, selecione SAP HANA.
   - Ao fornecer o URL do SAP HANA, forneça o URL da sua instância.

     Os URLs JDBC do SAP HANA estão no formato `jdbc:sap://`*`saphanaHostname`*`:`*`saphanaPort`*`/?`*`databaseName`*`=`*`saphanaDBname`*`,`*`Paramete`*

     O AWS Glue requer os seguintes parâmetros de URL do JDBC:

     - `databaseName`: um banco de dados padrão no SAP HANA ao qual se conectar.
   - Ao selecionar um Segredo da AWS, forneça o *`secretName`*.

Depois de criar uma conexão AWS Glue SAP HANA, siga estas etapas antes de executar seu trabalho do AWS Glue:

- Conceda ao perfil do IAM associada ao seu trabalho do AWS Glue permissão para ler *`secretName`*.

Criar um nó de origem do SAP HANA

Pré-requisitos necessários

- Uma conexão AWS Glue SAP HANA configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called “Criar uma conexão SAP HANA”.
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Uma tabela do SAP HANA da qual você gostaria de ler, *`tableName`*, ou consultar *`targetQuery`*.

  Uma tabela pode ser especificada com um nome de tabela e um nome de esquema do SAP HANA, no formulário *`schemaName`*`.`*`tableName`*. O nome do esquema e o separador "." não serão necessários se a tabela estiver no esquema padrão, "público". Chame isso de *`tableIdentifier`*. Observe que o banco de dados é fornecido como um parâmetro de URL do JDBC em `connectionName`.

Adicionar uma fonte de dados do SAP HANA

Para adicionar um nó de Fonte de dados – SAP HANA:

1. Escolha a conexão para sua fonte de dados do SAP HANA. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão com o SAP HANA. Para mais informações, consulte a seção the section called “Criar uma conexão SAP HANA” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Escolha uma opção de Fonte do SAP HANA:
   - Escolher uma única tabela: acesse todos os dados de uma única tabela.
   - Inserir consulta personalizada: acesse um conjunto de dados de várias tabelas com base em sua consulta personalizada.
3. Se você escolheu uma única tabela, insira *`tableName`*.

   Se você escolheu Inserir consulta personalizada, insira uma consulta SQL SELECT.
4. Em Propriedades personalizadas do SAP HANA, insira parâmetros e valores conforme necessário.

Criar um nó de destino do SAP HANA

Pré-requisitos necessários

- Uma conexão AWS Glue SAP HANA configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called “Criar uma conexão SAP HANA”.
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Uma tabela do SAP HANA na qual você gostaria de escrever, *`tableName`*.

  Uma tabela pode ser especificada com um nome de tabela e um nome de esquema do SAP HANA, no formulário *`schemaName`*`.`*`tableName`*. O nome do esquema e o separador "." não serão necessários se a tabela estiver no esquema padrão, "público". Chame isso de *`tableIdentifier`*. Observe que o banco de dados é fornecido como um parâmetro de URL do JDBC em `connectionName`.

Adicionar um destino de dados do SAP HANA

Para adicionar um nó de Destino de dados – SAP HANA:

1. Escolha a conexão para sua fonte de dados do SAP HANA. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão com o SAP HANA. Para mais informações, consulte a seção the section called “Criar uma conexão SAP HANA” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Configure o nome da tabela fornecendo o *`tableName`*.
3. Em Propriedades personalizadas do Teradata, insira parâmetros e valores conforme necessário.

Opções avançadas

Você pode fornecer opções avançadas ao criar um nó do SAP HANA. Essas opções são as mesmas disponíveis ao programar AWS Glue para scripts do Spark.

Consulte the section called “Conexões do SAP HANA”.

Conectar ao Snowflake no AWS Glue Studio

> **Note**
>
> Você pode usar o AWS Glue para Spark para ler e gravar em tabelas no Snowflake no AWS Glue 4.0 e versões posteriores. Para configurar uma conexão Snowflake com trabalhos AWS Glue programaticamente, consulte Conexões do Redshift.

O AWS Glue fornece suporte integrado para o Snowflake. O AWS Glue Studio fornece uma interface visual para conectar ao Snowflake, criar trabalhos de integração de dados e executá-los no runtime do AWS Glue Studio Spark sem servidor.

Tópicos

Criar uma conexão com o Snowflake

Ao adicionar um nó Fonte de dados - SnowflakeAWS Glue Studio, você pode escolher uma conexão existente do AWS Glue Snowflake ou criar uma nova conexão. Você deve escolher um tipo de conexão do `SNOWFLAKE` e não um tipo de conexão `JDBC` configurada para se conectar ao Snowflake. Faça o seguinte procedimento para criar uma conexão com o AWS Glue Snowflake:

Criar uma conexão com o Snowflake

1. No Snowflake, gere um usuário, *`SnowflakeUser`* e senha, *`snowflakePassword`*.
2. *`Determine com qual armazém do Snowflake esse usuário interagirá, SnowflakeWarehouse`*. Defina-o como o `DEFAULT_WAREHOUSE` para *`SnowFlakeUser`* no Snowflake ou lembre-se dele para a próxima etapa.
3. No AWS Secrets Manager, crie um segredo usando suas credenciais do Snowflake. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criação de um segredo do AWS Secrets Manager na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *`SecretName`*, para a próxima etapa.

- Ao selecionar pares de chave/valor, crie um par para *snowflakeUser* com a chave `sfUser`.
- Ao selecionar pares de chave/valor, crie um par para *snowflakePassword* com a chave `sfPassword`.
- Ao selecionar pares de chave/valor, crie um par para *snowflakeWarehouse* com a chave `sfWarehouse`. Isso não é necessário se for definido um padrão no Snowflake.

4. No catálogo de dados do AWS Glue, crie uma conexão seguindo as etapas em Adicionar uma conexão do AWS Glue. Depois de criar a conexão, guarde o nome da conexão, *connectionName*, para a próxima etapa.

   - Ao selecionar um Tipo de conexão, selecione Snowflake.
   - Ao selecionar a URL do Snowflake, forneça a URL da sua instância do Snowflake. A URL usará um nome de host no formulário *account_identifier*`.snowflakecomputing.com`.
   - Ao selecionar um Segredo da AWS, forneça o *secretName*.

Criar um nó de origem do Snowflake

Permissões necessárias

Trabalhos do AWS Glue Studio usando fontes de dados do Snowflake exigem permissões adicionais. Para obter mais informações sobre como adicionar permissões a trabalhos de ETL, consulte Review IAM permissions needed for ETL jobs.

As conexões AWS Glue do `SNOWFLAKE` usam um segredo do AWS Secrets Manager para fornecer informações de credenciais. Os perfis de pré-visualização de trabalhos e dados no AWS Glue Studio devem ter permissão para ler esse segredo.

Adicionar uma fonte de dados do Snowflake

Pré-requisitos:

- Um segredo do AWS Secrets Manager para suas credenciais do Snowflake
- Uma conexão de catálogo de dados do AWS Glue do tipo Snowflake

Para adicionar um nó fonte de dados: Snowflake:

1. Escolha a conexão para sua fonte de dados do Snowflake. Isso pressupõe que a conexão já exista e que você possa selecionar entre as conexões existentes. Se precisar criar uma

conexão, escolha Criar conexão com o Snowflake. Para obter mais informações, consulte Visão geral do uso de conectores e conexões.

Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades. As informações sobre a conexão estão visíveis, incluindo URL, grupos de segurança, sub-rede, zona de disponibilidade, descrição e timestamps de criação (UTC) e última atualização (UTC).

2. Escolha uma opção de fonte do Snowflake:

   - Escolher uma única tabela: essa é a tabela que contém os dados que você deseja acessar de uma única tabela do Snowflake.
   - Inserir consulta personalizada: permite que você acesse um conjunto de dados de várias tabelas do Snowflake com base na sua consulta personalizada.

3. Se você escolheu uma única tabela, insira o nome de um esquema do Snowflake.

   Ou escolha Inserir consulta personalizada. Escolha essa opção para acessar um conjunto de dados personalizado de várias tabelas do Snowflake. Ao escolher essa opção, insira a consulta do Snowflake.

4. Nas opções de Desempenho e segurança (opcional),

   - Ative o envio de consultas: escolha se você deseja transferir o trabalho para a instância do Snowflake.

5. Em Propriedades personalizadas do Snowflake (opcional), insira parâmetros e valores conforme necessário.

Criar um nó de destino do Snowflake

Permissões necessárias

Trabalhos do AWS Glue Studio usando fontes de dados do Snowflake exigem permissões adicionais. Para obter mais informações sobre como adicionar permissões a trabalhos de ETL, consulte Review IAM permissions needed for ETL jobs.

As conexões AWS Glue do `SNOWFLAKE` usam um segredo do AWS Secrets Manager para fornecer informações de credenciais. Os perfis de pré-visualização de trabalhos e dados no AWS Glue Studio devem ter permissão para ler esse segredo.

Adicionar um destino de dados do Snowflake

Para criar um nó de destino do Snowflake:

1. Escolha uma tabela existente do Snowflake como destino ou insira um novo nome de tabela.
2. Ao usar o nó de destino Nó de destino: Snowflake, você pode escolher entre as seguintes opções:

   - APPEND: se uma tabela já existir, despejar todos os novos dados na tabela como uma inserção. Se a tabela não existir, criar e inserir todos os novos dados.
   - MERGE: o AWS Glue atualizará ou anexará dados à sua tabela de destino com base nas condições que você especificar.

     Escolha as opções:

     - Escolher chaves e ações simples: escolha as colunas a serem usadas como chaves de correspondência entre os dados de origem e seu conjunto de dados de destino.

       Especifique as seguintes opções quando correspondidas:

       - Atualizar o registro em seu conjunto de dados de destino com os dados da fonte.
       - Excluir o registro em seu conjunto de dados de destino.

       Especifique as seguintes opções quando não correspondidas:

       - Inserir os dados de origem como uma nova linha em seu conjunto de dados de destino.
       - Não executar nenhuma ação.
     - Inserir declaração MERGE personalizada: você pode então escolher Validar declaração de Merge para verificar se a declaração é válida ou inválida.
   - TRUNCATE: se uma tabela já existir, truncar os dados da tabela limpando primeiro o conteúdo da tabela de destino. Se o truncamento for bem-sucedido, inserir todos os dados. Se a tabela não existir, criar a tabela e inserir todos os novos dados. Se o truncamento não tiver sucesso, a operação falhará.
   - DROP: se uma tabela já existir, excluir os metadados e os dados da tabela. Se a exclusão for bem-sucedida, inserir todos os dados. Se a tabela não existir, criar a tabela e inserir todos os novos dados. Se o drop não tiver sucesso, a operação falhará.

Opções avançadas

Consulte as Snowflake connections no guia do desenvolvedor do AWS Glue.

Conectar ao Teradata Vantage no AWS Glue Studio

O AWS Glue oferece suporte integrado ao Teradata Vantage. O AWS Glue Studio fornece uma interface visual para conectar ao Teradata, criar trabalhos de integração de dados e executá-los no runtime Spark do AWS Glue Studio sem servidor.

Tópicos

Criar uma conexão Teradata Vantage

Para se conectar ao Teradata Vantage via AWS Glue, será necessário criar e armazenar suas credenciais do Teradata em um segredo do AWS Secrets Manager e, em seguida, associar esse segredo a uma conexão ao AWS Glue do Teradata.

Pré-requisitos:

- Se você estiver acessando seu ambiente Teradata via Amazon VPC, configure a Amazon VPC para permitir que seu trabalho do AWS Glue se comunique com o ambiente Teradata. Recomendamos não acessar o ambiente Teradata via Internet pública.

  Na Amazon VPC, identifique ou crie uma VPC, uma Sub-rede e um Grupo de segurança que o AWS Glue usará durante a execução do trabalho. Além disso, você precisa garantir que a Amazon VPC esteja configurada para permitir o tráfego de rede entre sua instância do Teradata e esse local. Seu trabalho precisará estabelecer uma conexão TCP com a porta cliente do Teradata. Para obter mais informações sobre portas do Teradata, consulte a Documentação do Teradata.

  Com base no layout da sua rede, a conectividade segura da VPC pode exigir alterações na Amazon VPC e em outros serviços de rede. Para obter mais informações sobre conectividade com a AWS, consulte Opções de conectividade da AWS na documentação da Teradata.

Para configurar uma conexão AWS Glue Teradata:

1. Em sua configuração do Teradata, identifique ou crie um usuário e a senha com os quais o AWS Glue se conectará, *teradataUser* e *teradataPassword*. Para obter mais informações, consulte Visão geral da segurança do Vantage na documentação do Teradata.
2. No AWS Secrets Manager, crie um segredo usando suas credenciais do Teradata. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *SecretName*, para a próxima etapa.

   - Ao selecionar Pares chave/valor, crie um par para a chave `user` com o valor *teradataUsername*.
   - Ao selecionar Pares chave/valor, crie um par para a chave `password` com o valor *teradataPassword*.
3. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *connectionName*, para a próxima etapa.

   - Ao selecionar um Tipo de conexão, selecione Teradata.
   - Ao fornecer o URL do JDBC, forneça o URL da sua instância. Você também pode codificar certos parâmetros de conexão separados por vírgula no URL do JDBC. O URL deve estar de acordo com o seguinte formato: `jdbc:teradata://`*teradataHostname*`/`*ParameterName*`=`*ParameterValue*`,`*ParameterName*

     Os parâmetros de URL compatíveis incluem:

     - `DATABASE`: o nome do banco de dados no host a ser acessado por padrão.
     - `DBS_PORT`: a porta do banco de dados usada para execução em uma porta não padrão.
   - Ao selecionar um Tipo de credencial, selecione AWS Secrets Manager e defina Segredo da AWS como *secretName*.
4. Nas seguintes situações, configurações adicionais podem ser necessárias:

   - Para instâncias do Teradata hospedadas na AWS em uma Amazon VPC
     - Será necessário fornecer informações de conexão da Amazon VPC à conexão do AWS Glue que define suas credenciais de segurança do Teradata. Ao criar ou atualizar sua conexão, defina VPC, Sub-rede e Grupos de segurança em Opções de rede.

Criar um nó de origem do Teradata

Pré-requisitos necessários

- Uma conexão AWS Glue Teradata Vantage configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called “Criar uma conexão Teradata Vantage”.
- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Uma tabela do Teradata da qual você gostaria de ler, *`tableName`*, ou consultar *`targetQuery`*.

Adicionar uma fonte de dados do Teradata

Para adicionar um nó de Fonte de dados – Teradata:

1. Escolha a conexão para sua fonte de dados do Teradata. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão. Para mais informações, consulte a seção the section called “Criar uma conexão Teradata Vantage” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Escolha uma opção de Fonte de dados teradata:

   - Escolher uma única tabela: acesse todos os dados de uma única tabela.
   - Inserir consulta personalizada: acesse um conjunto de dados de várias tabelas com base em sua consulta personalizada.
3. Se você escolheu uma única tabela, insira *`tableName`*.

   Se você escolheu Inserir consulta personalizada, insira uma consulta SQL SELECT.
4. Em Propriedades personalizadas do Teradata, insira parâmetros e valores conforme necessário.

Criar um nó de destino do Teradata

Pré-requisitos necessários

- Uma conexão AWS Glue Teradata Vantage configurada com um segredo do AWS Secrets Manager, conforme descrito na seção anterior, the section called “Criar uma conexão Teradata Vantage”.

- Permissões apropriadas em seu trabalho para ler o segredo usado pela conexão.
- Uma tabela do Teradata na qual você gostaria de escrever, *tableName*.

Adicionar um destino de dados do Teradata

Para adicionar um nó de Destino de dados – Teradata:

1. Escolha a conexão para sua fonte de dados do Teradata. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão com o Teradata. Para obter mais informações, consulte Visão geral do uso de conectores e conexões.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Configure o nome da tabela fornecendo o *tableName*.
3. Em Propriedades personalizadas do Teradata, insira parâmetros e valores conforme necessário.

Opções avançadas

Você pode fornecer opções avançadas ao criar um nó do Teradata. Essas opções são as mesmas disponíveis ao programar AWS Glue para scripts do Spark.

Consulte the section called “Conexões do Teradata Vantage”.

Conectar ao Vertica no AWS Glue Studio

O AWS Glue oferece suporte integrado ao Vertica. O AWS Glue Studio fornece uma interface visual para conectar ao Vertica, criar trabalhos de integração de dados e executá-los no runtime Spark do AWS Glue Studio sem servidor.

Tópicos

- Criar uma conexão Vertica
- Criar um nó de origem do Vertica
- Criar um nó de destino do Vertica
- Opções avançadas

Criar uma conexão Vertica

Pré-requisitos:

- Um bucket ou uma pasta do Amazon S3 para usar como armazenamento temporário ao ler e escrever no banco de dados, referido por *tempS3Path*.

> **Note**
>
> Quando o Vertica é usado em pré-visualizações de dados de trabalhos do AWS Glue, os arquivos temporários podem não ser removidos automaticamente de *tempS3Path*. Para garantir a remoção de arquivos temporários, encerre diretamente a sessão de visualização de dados escolhendo Encerrar sessão no painel Visualização de dados.
> Se não for possível garantir que a sessão de visualização de dados seja encerrada diretamente, considere definir a configuração do ciclo de vida do Amazon S3 para remover dados antigos. Recomendamos remover dados com mais de 49 horas com base no runtime máximo do trabalho somado a uma margem. Para obter mais informações sobre a configuração do Amazon S3, consulte Gerenciar o ciclo de vida do armazenamento na documentação do Amazon S3.

- Uma política do IAM com permissões apropriadas para seu caminho do Amazon S3 que você pode associar ao seu perfil de trabalho do AWS Glue.
- Se a sua instância do Vertica estiver em uma Amazon VPC, configure a Amazon VPC para permitir que seu trabalho do AWS Glue se comunique com a instância do Vertica sem que o tráfego passe pela Internet pública.

  Na Amazon VPC, identifique ou crie uma VPC, uma Sub-rede e um Grupo de segurança que o AWS Glue usará durante a execução do trabalho. Além disso, você precisa garantir que a Amazon VPC esteja configurada para permitir o tráfego de rede entre sua instância do Vertica e esse local. Seu trabalho precisará estabelecer uma conexão TCP com a porta cliente do Vertica (por padrão, 5433). Com base no layout da rede, isso pode exigir alterações em regras do grupo de segurança, ACLs de rede, gateways de NAT e conexões de emparelhamento.

Para configurar uma conexão com o Vertica:

1. No AWS Secrets Manager, crie um segredo usando suas credenciais do Vertica, *verticaUsername* e *verticaPassword*. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS

Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *SecretName*, para a próxima etapa.

- Ao selecionar Pares chave/valor, crie um par para a chave `user` com o valor *verticaUsername*.
- Ao selecionar Pares chave/valor, crie um par para a chave `password` com o valor *verticaPassword*.

2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *connectionName*, para a próxima etapa.

   - Ao selecionar um Tipo de conexão, selecione Vertica.
   - Ao selecionar Host do Vertica, forneça o URL da sua instalação do Vertica.
   - Ao selecionar Porta do Vertica, a porta pela qual sua instalação do Vertica está disponível.
   - Ao selecionar um Segredo da AWS, forneça o *secretName*.

3. Nas seguintes situações, configurações adicionais podem ser necessárias:

   - Para instâncias do Teradata hospedadas na AWS em uma Amazon VPC
     - Forneça informações de conexão da Amazon VPC à conexão do AWS Glue que define suas credenciais de segurança do Vertica. Ao criar ou atualizar sua conexão, defina VPC, Sub-rede e Grupos de segurança em Opções de rede.

Será necessário realizar as seguintes etapas antes de executar seu trabalho do AWS Glue:

- Conceda ao perfil do IAM associado ao seu trabalho do AWS Glue permissão para *tempS3Path*.
- Conceda ao perfil do IAM associada ao seu trabalho do AWS Glue permissão para ler *secretName*.

Criar um nó de origem do Vertica

Pré-requisitos necessários

- Uma conexão de Catálogo de Dados AWS Glue do tipo Vertica, *connectionName* e um local temporário do Amazon S3, *temps3Path*, conforme descrito na seção the section called "Criar uma conexão Vertica" anterior.
- Uma tabela do Vertica da qual você gostaria de ler, *tableName*, ou consultar *targetQuery*.

Adicionar uma fonte de dados do Vertica

Para adicionar um nó de Fonte de dados – Vertica:

1. Escolha a conexão para sua fonte de dados do Vertica. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão do Vertica. Para mais informações, consulte a seção the section called “Criar uma conexão Vertica” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Escolha o Banco de dados que contém sua tabela.
3. *Escolha a **Área de preparação no Amazon S3** e insira um URI do S3A para `tempS3Path`*.
4. Escolha a Fonte do Vertica.

   - Escolher uma única tabela: acesse todos os dados de uma única tabela.
   - Inserir consulta personalizada: acesse um conjunto de dados de várias tabelas com base em sua consulta personalizada.
5. Se você escolheu uma única tabela, insira *`tableName`* e, opcionalmente, selecione um Esquema.

   Se você escolheu Inserir consulta personalizada, insira uma consulta SQL SELECT e, opcionalmente, selecione um Esquema.
6. Em Propriedades personalizadas do Vertica, insira parâmetros e valores conforme necessário.

Criar um nó de destino do Vertica

Pré-requisitos necessários

- Uma conexão de Catálogo de Dados AWS Glue do tipo Vertica, *`connectionName`* e um local temporário do Amazon S3, *`temps3Path`*, conforme descrito na seção the section called “Criar uma conexão Vertica” anterior.

Adicionar um destino de dados do Vertica

Para adicionar um nó de Destino de dados – Vertica:

1. Escolha a conexão para sua fonte de dados do Vertica. Como você o criou, ele deve estar disponível no menu suspenso. Se precisar criar uma conexão, escolha Criar conexão do Vertica. Para mais informações, consulte a seção the section called “Criar uma conexão Vertica” anterior.

   Depois de escolher uma conexão, você pode visualizar as propriedades da conexão clicando em Exibir propriedades.
2. Escolha o Banco de dados que contém sua tabela.
3. *Escolha a **Área de preparação no Amazon S3** e insira um URI do S3A para tempS3Path*.
4. Insira *tableName* e, opcionalmente, selecione um Esquema.
5. Em Propriedades personalizadas do Vertica, insira parâmetros e valores conforme necessário.

Opções avançadas

Você pode fornecer opções avançadas ao criar um nó do Vertica. Essas opções são as mesmas disponíveis ao programar AWS Glue para scripts do Spark.

Consulte the section called “Conexões do Vertica”.

Usar conectores e conexões com o AWS Glue Studio

O AWS Glue oferece suporte integrado aos armazenamentos de dados usados com frequência (como Amazon Redshift, Amazon Aurora, Microsoft SQL Server, MySQL, MongoDB e PostgreSQL) usando conexões JDBC. O AWS Glue também permite usar drivers do JDBC personalizados em seus trabalhos de extração, transformação e carregamento (ETL). Para armazenamentos de dados que não são suportados nativamente, como aplicações SaaS, você pode usar conectores.

Um conector é um pacote de código opcional que ajuda a acessar armazenamentos de dados no AWS Glue Studio. Você pode assinar vários conectores oferecidos no AWS Marketplace.

Ao criar trabalhos de ETL, você pode usar um armazenamento de dados com suporte nativo, um conector de AWS Marketplace ou seus próprios conectores personalizados. Se você usar um conector, primeiro deve criar uma conexão para ele. Uma conexão que contém as propriedades

necessárias para se conectar a um datastore específico. Você usa a conexão com suas origens e destinos de dados no trabalho de ETL. Conectores e conexões trabalham juntos para facilitar o acesso aos armazenamentos de dados.

Tópicos

Visão geral do uso de conectores e conexões

Uma conexão que contém as propriedades necessárias para se conectar a um datastore específico. Ao criar uma conexão, ela é armazenada no AWS Glue Data Catalog. Você escolhe um conector e, em seguida, cria uma conexão com base nele.

Você pode assinar conectores para armazenamentos de dados sem suporte nativo e AWS Marketplace, em seguida, usar esses conectores ao criar conexões. Os desenvolvedores também podem criar seus próprios conectores, e você pode usá-los ao criar conexões.

> **Note**
>
> As conexões criadas usando AWS Marketplace conectores personalizados ou em AWS Glue Studio aparecem no AWS Glue console com o tipo definido como. `UNKNOWN`

As etapas a seguir descrevem o processo geral do uso de conectores no AWS Glue Studio:

1. Assine um conector em AWS Marketplace, ou desenvolva seu próprio conector e faça o upload paraAWS Glue Studio. Para ter mais informações, consulte Adição de conectores ao AWS Glue Studio.

2. Revise as informações de uso do conector. Você pode encontrar essas informações na guia Usage (Uso) na página de produto do conector. Por exemplo, se você clicar na guia Uso desta página do produto, AWS GlueConnector for Google BigQuery, poderá ver na seção Recursos adicionais um link para um blog sobre o uso desse conector. Outros conectores podem conter links para instruções na seção Visão geral, como mostrado na página de produto do Conector do Cloudwatch Logs para AWS Glue.
3. Crie uma conexão. Você escolhe qual conector usar e fornece informações adicionais para a conexão, como credenciais de login, strings de URI e informações da nuvem privada virtual (VPC). Para ter mais informações, consulte Criar conexões para conectores.
4. Crie uma função do IAM para o seu trabalho. O trabalho assume a permissão de IAM role que você especificou ao criá-lo. Essa função do IAM precisa ter as permissões obrigatórias para autenticar com, extrair dados de e gravar dados em seus armazenamentos de dados.
5. Crie um trabalho de ETL e configure as propriedades da origem dos dados para o seu trabalho de ETL. Forneça as opções de conexão e as informações de autenticação conforme instruído pelo provedor de conector personalizado. Para ter mais informações, consulte Criação de trabalhos com conectores personalizados.
6. Personalize seu trabalho de ETL adicionando transformações ou armazenamentos de dados adicionais, conforme descrito em Visual ETL com AWS Glue Studio.
7. Se estiver usando um conector para o destino de dados, configure as propriedades de destino de dados para seu trabalho de ETL. Forneça as opções de conexão e as informações de autenticação conforme instruído pelo provedor de conector personalizado. Para ter mais informações, consulte the section called “Criação de trabalhos com conectores personalizados”.
8. Personalize o ambiente de execução de trabalho configurando as propriedades do trabalho, conforme descrito em Modificar as propriedades do trabalho.
9. Execute o trabalho.

Adição de conectores ao AWS Glue Studio

Um conector é um pedaço de código que facilita a comunicação entre o datastore e o AWS Glue. Você pode assinar um conector oferecido em AWS Marketplace ou criar seu próprio conector personalizado.

Tópicos

- Inscrevendo-se em conectores AWS Marketplace
- Criar conectores personalizados

Inscrevendo-se em conectores AWS Marketplace

AWS Glue Studiofacilita a adição de conectores de AWS Marketplace.

Para adicionar um conector de AWS Marketplace a AWS Glue Studio

1. No console do AWS Glue Studio, escolha Connectors (Conectores), no painel de navegação do console.
2. Na página Conectores, escolha Ir para o AWS Marketplace.
3. Em AWS Marketplace, em Produtos em destaque, escolha o conector que você deseja usar. Você pode escolher um dos conectores em destaque ou usar a pesquisa. Você pode pesquisar o nome ou tipo de conector e usar opções para refinar os resultados da pesquisa.

   Se você desejar usar um dos conectores em destaque, escolha View product (Ver produto). Se você usou a pesquisa para localizar um conector, escolha o nome do conector.
4. Na página de produto do conector, use as guias para visualizar informações sobre o conector. Se você decidir adquirir esse conector, escolha Continue to Subscribe (Continuar para assinar).
5. Forneça as informações de pagamento e escolha Continue to Configure (Continuar para configurar).
6. Na página Configure this software (Configurar este software), escolha o método de implantação e a versão do conector a serem usados. Em seguida, escolha Continue to Launch (Continuar para iniciar).
7. Na página Lauch this software (Iniciar este software), você pode analisar as Usage Instructions (Instruções de uso) fornecidas pelo provedor do conector. Quando estiver pronto para continuar, escolha Ativar conexão no AWS Glue Studio.

   Após um curto período, o console apresenta a pagina Create marketplace connection (Criar conexão com o marketplace) no AWS Glue Studio.
8. Crie uma conexão que use esse conector, conforme descrito em Criar conexões para conectores.

   Como alternativa, você pode escolher Activate connector only (Ativar somente conector) para ignorar a criação de uma conexão no momento. Você deve criar uma conexão em uma data posterior para que possa usar o conector.

Criar conectores personalizados

Você também pode construir seu próprio conector e depois carregar o código do conector no AWS Glue Studio.

Os conectores personalizados são integrados ao AWS Glue Studio pela API de runtime do Spark no AWS Glue. O runtime do Spark no AWS Glue permite conectar qualquer conector compatível com a interface do Spark, Athena ou JDBC. Ele permite que você insira qualquer opção de conexão que esteja disponível com o conector personalizado.

Você pode encapsular todas as suas propriedades de conexão com conexões do AWS Glue e fornecer o nome da conexão ao seu trabalho de ETL. A integração com conexões do Data Catalog permite que você use as mesmas propriedades de conexão em várias chamadas em uma única aplicação Spark ou em diferentes aplicações.

Você pode especificar opções adicionais para a conexão. O script de trabalho que o AWS Glue Studio gera contém uma entrada `Datasource` que usa a conexão para ligar o conector com as opções de conexão especificadas. Por exemplo:

```
Datasource = glueContext.create_dynamic_frame.from_options(connection_type =
"custom.jdbc", connection_options = {"dbTable":"Account","connectionName":"my-custom-
jdbc-
connection"}, transformation_ctx = "DataSource0")
```

Para adicionar um conector personalizado ao AWS Glue Studio

1. Crie o código para o seu conector personalizado. Para ter mais informações, consulte Desenvolvimento de conectores personalizados.
2. Adicione suporte ao seu conector para os recursos do AWS Glue. Aqui estão alguns exemplos desses recursos e como eles são usados dentro do script de trabalho gerado pelo AWS Glue Studio:

   - Data type mapping (Mapeamento de tipo de dados): seu conector pode converter os tipos das colunas ao lê-las a partir do datastore subjacente. Por exemplo, um `dataTypeMapping` de `{"INTEGER":"STRING"}` converte todas as colunas do tipo `Integer` para colunas do tipo `String` ao analisar os registros e construir o `DynamicFrame`. Isso ajuda os usuários a converter colunas para tipos de sua escolha.

     ```
     DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type
     = "custom.jdbc", connection_options = {"dataTypeMapping":{"INTEGER":"STRING"}",
     ```

```
connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
```

- Partitioning for parallel reads (Particionamento para leituras paralelas): o AWS Glue permite leituras de dados paralelas do datastore ao particionar os dados em uma coluna. Você deve especificar a coluna de partição, o limite de partição inferior, o limite de partição superior e o número de partições. Esse recurso permite que você faça uso do paralelismo de dados e de vários executores do Spark alocados para a aplicação do 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")
```

- Use AWS Secrets Manager para armazenar credenciais — A conexão do Catálogo de Dados também pode conter um `secretId` para um segredo armazenado em AWS Secrets Manager. O AWS segredo pode armazenar com segurança as informações de autenticação e credenciais e fornecê-las em tempo de execução. AWS Glue Como alternativa, você pode especificar o `secretId` no script do Spark da seguinte forma:

```
DataSource = glueContext.create_dynamic_frame.from_options(connection_type
= "custom.jdbc", connection_options = {"connectionName":"test-connection-jdbc",
 "secretId"-> "my-secret-id"}, transformation_ctx = "DataSource0")
```

- Filtrando os dados de origem com predicados de linha e projeções de coluna: o runtime do Spark no AWS Glue também permite que os usuários propaguem consultas SQL para filtrar dados na origem com predicados de linha e projeções de coluna. Isso permite que seu trabalho de ETL carregue dados filtrados mais rapidamente a partir de armazenamentos de dados que suportam propagações. Uma consulta SQL de exemplo enviada para uma origem dos dados JDBC é: `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 trabalho: o AWS Glue suporta carregamento incremental de dados de fontes JDBC. O AWS Glue mantém o controle do último registro processado do datastore e processa novos registros de dados nas execuções de trabalho de ETL subsequentes. Os marcadores

de trabalho usam a chave primária como a coluna padrão para a chave do marcador, desde que essa coluna aumente ou diminua sequencialmente. Para obter mais informações sobre marcadores de trabalhos, consulte Marcadores de trabalho no Guia do desenvolvedor do 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. Empacote o conector personalizado como um arquivo JAR e carregue-o no Amazon S3.
4. Teste seu conector personalizado. Para obter mais informações, consulte as instruções GitHub em Glue Custom Connectors: Local Validation Tests Guide.
5. No console do AWS Glue Studio, escolha Connectors (Conectores), no painel de navegação do console.
6. Na página Connectors (Conectores) escolha Create custom connector (Criar um conector personalizado).
7. Na página Create custom connector (Criar um conector personalizado), insira as informações a seguir:

   - O caminho para o local do arquivo JAR do código personalizado no Amazon S3.
   - Um nome para o conector que será usado pelo AWS Glue Studio.
   - Seu tipo de conector, que pode ser JDBC, Spark ou Athena.
   - O nome do ponto de entrada dentro do código personalizado que o AWS Glue Studio chama para usar o conector.
     - Para conectores JDBC, esse campo deve ser o nome da classe do driver do JDBC.
     - Para conectores do Spark, esse campo deve ser o nome da classe da origem dos dados totalmente qualificada, ou seu alias, que você usa ao carregar a origem dos dados do Spark com o operador `format`.
   - (Somente JDBC) o URL base usado pela conexão JDBC para o datastore.
   - (Opcional) uma descrição do conector personalizado.
8. Escolha Create connector (Criar conector).
9. Na página Connectors (Conectores), crie uma conexão que use esse conector, conforme descrito em Criar conexões para conectores.

Conexões disponíveis

As conexões a seguir estão disponíveis ao criar conexões para conectores:

- Amazon Aurora: um mecanismo de banco de dados relacional escalável e de alta performance com segurança, backup e restauração integrados e aceleração na memória.
- Amazon DocumentDB: um serviço de banco de dados de documentos escalável, altamente disponível e totalmente gerenciado que oferece suporte às APIs do MongoDB e SQL.
- Amazon Redshift: um serviço de banco de dados de documentos escalável, altamente disponível e totalmente gerenciado que oferece suporte às APIs do MongoDB e SQL.
- Azure SQL: um serviço de banco de dados relacional baseado em nuvem do Microsoft Azure que fornece recursos de armazenamento e gerenciamento de dados escaláveis, confiáveis e seguros.
- Cosmos DB: um serviço de banco de dados em nuvem distribuído globalmente do Microsoft Azure que fornece recursos de consulta e armazenamento de dados escaláveis e de alta performance.
- Google BigQuery — um data warehouse em nuvem sem servidor para executar consultas SQL rápidas em grandes conjuntos de dados.
- JDBC: um sistema de gerenciamento de banco de dados relacional (RDBMS) que usa uma API Java para se conectar e interagir com conexões de dados.
- Kafka: uma plataforma de processamento de fluxo de código aberto usada para streaming de dados e mensagens em tempo real.
- MariaDB: um fork do MySQL desenvolvido pela comunidade que oferece performance, escalabilidade e recursos aprimorados.
- MongoDB: um banco de dados multiplataforma orientado a documentos que fornece alta escalabilidade, flexibilidade e performance.
- MongoDB Atlas: uma oferta de banco de dados como serviço (DBaaS) baseada em nuvem do MongoDB que simplifica o gerenciamento e o dimensionamento das implantações do MongoDB.
- Microsoft SQL Server: um sistema de gerenciamento de banco de dados relacional (RDBMS) da Microsoft que fornece recursos robustos de armazenamento, análise e geração de relatórios de dados.
- MySQL: um sistema de gerenciamento de banco de dados relacional (RDBMS) de código aberto que é amplamente usado em aplicações Web e é conhecido por sua confiabilidade e escalabilidade.
- Rede: uma fonte de dados de rede representa um recurso ou serviço acessível pela rede que pode ser acessado por uma plataforma de integração de dados.

- OpenSearch— uma fonte de OpenSearch dados é um aplicativo que OpenSearch pode se conectar e ingerir dados de.
- Oracle: um sistema de gerenciamento de banco de dados relacional (RDBMS) da Oracle Corporation que fornece recursos robustos de armazenamento, análise e geração de relatórios de dados.
- PostgreSQL: um sistema de gerenciamento de banco de dados relacional (RDBMS) de código aberto que fornece recursos robustos de armazenamento, análise e geração de relatórios de dados.
- Salesforce — A Salesforce fornece software de gerenciamento de relacionamento com o cliente (CRM) que ajuda você com vendas, atendimento ao cliente, comércio eletrônico e muito mais. Se você é usuário do Salesforce, pode se conectar AWS Glue à sua conta do Salesforce. Em seguida, você pode usar o Salesforce como fonte de dados ou destino em suas tarefas de ETL. Execute essas tarefas para transferir dados entre o Salesforce e AWS os serviços ou outros aplicativos compatíveis.
- SAP HANA: uma plataforma de análise e banco de dados na memória que fornece processamento rápido de dados, análises avançadas e integração de dados em tempo real.
- Snowflake: um data warehouse baseado em nuvem que fornece serviços de análise e armazenamento de dados escaláveis e de alta performance.
- Teradata: um sistema de gerenciamento de banco de dados relacional (RDBMS) que fornece recursos de armazenamento, análise e geração de relatórios de dados de alta performance.
- Vertica: um data warehouse analítico orientado por colunas projetado para análise de big data que oferece performance rápida de consultas, análises avançadas e escalabilidade.

Criar conexões para conectores

Uma conexão do AWS Glue é um objeto do Data Catalog que armazena informações da conexão para determinado armazenamento de dados. As conexões armazenam credenciais de login, strings do URI, informações da nuvem privada virtual (VPC) e muito mais. A criação de conexões no Data Catalog economiza o esforço de ter que especificar todos os detalhes da conexão toda vez que um trabalho é criado.

Para criar uma conexão para um conector

1. No console do AWS Glue Studio, escolha Connectors (Conectores), no painel de navegação do console. Na seção Conexões de saída, escolha Criar conexão.

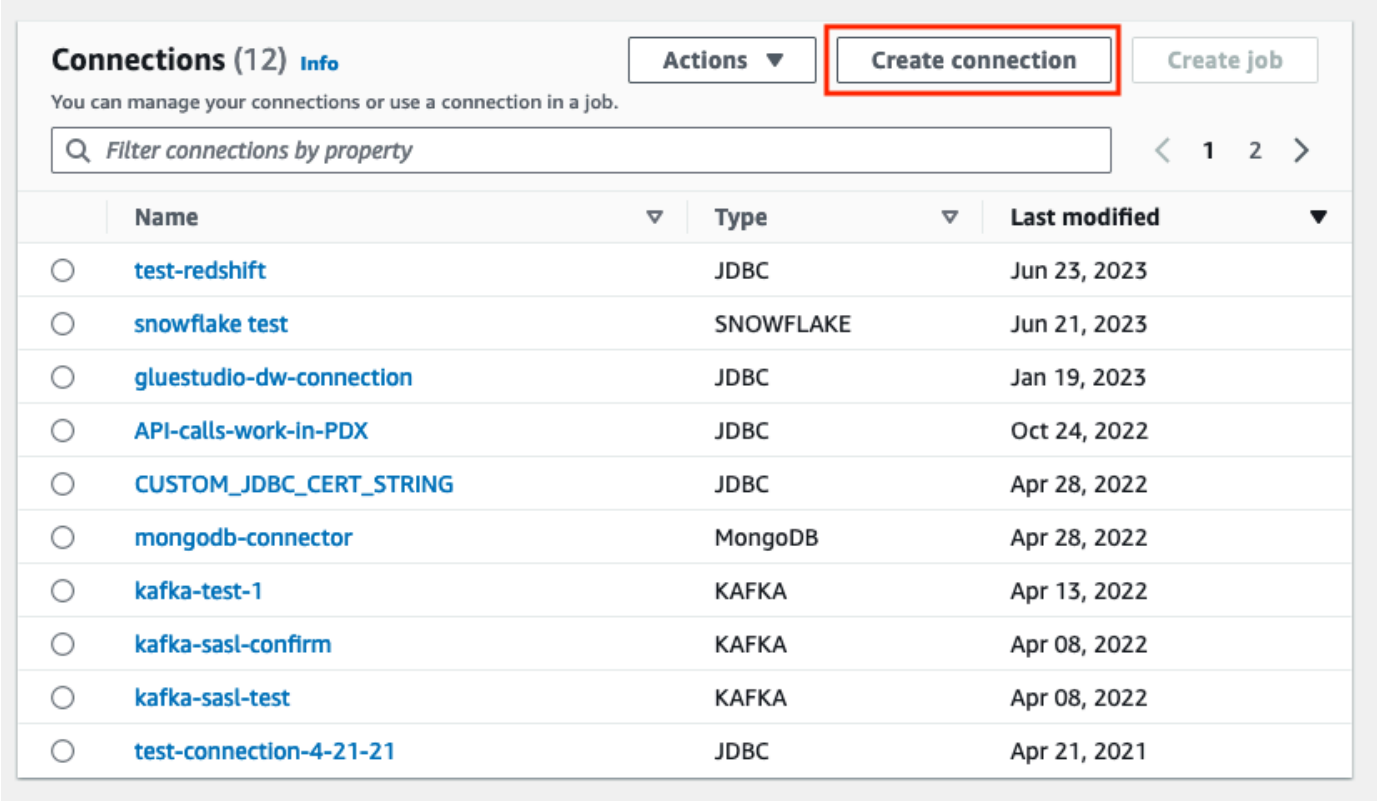
2. Escolha a fonte de dados para a qual deseja criar uma conexão na etapa 1 do assistente Criar conexão de dados. Há várias maneiras de visualizar as fontes de dados disponíveis, como:

   - Filtre as fontes de dados disponíveis escolhendo uma guia. Por padrão, Todos os conectores está selecionado.
   - Alterne a Lista para visualizar as fontes de dados como uma lista ou volte para a Grade para ver os conectores disponíveis no layout de grade.
   - Use a barra de pesquisa para restringir a lista de fontes de dados. À medida que você digita, as correspondências de pesquisa são exibidas e as fontes não correspondentes são removidas da exibição.

   Depois de escolher a fonte de dados, escolha Avançar.

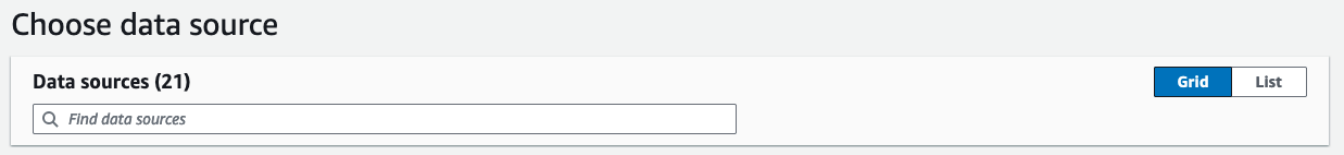
3. Configure a conexão na Etapa 2 do assistente.

   Insira os detalhes da conexão. Dependendo do tipo de conector selecionado, será solicitado que você insira informações adicionais:



4. Escolha a fonte de dados para a qual deseja criar uma conexão na etapa 1 do assistente Criar conexão de dados. Há várias maneiras de visualizar as fontes de dados disponíveis. Por padrão, você verá todas as fontes de dados disponíveis em um layout de grade. Você também pode:

- Alterne a Lista para visualizar as fontes de dados como uma lista ou volte para a Grade para ver os conectores disponíveis no layout de grade.
- Use a barra de pesquisa para restringir a lista de fontes de dados. À medida que você digita, as correspondências de pesquisa são exibidas e as fontes não correspondentes são removidas da exibição.

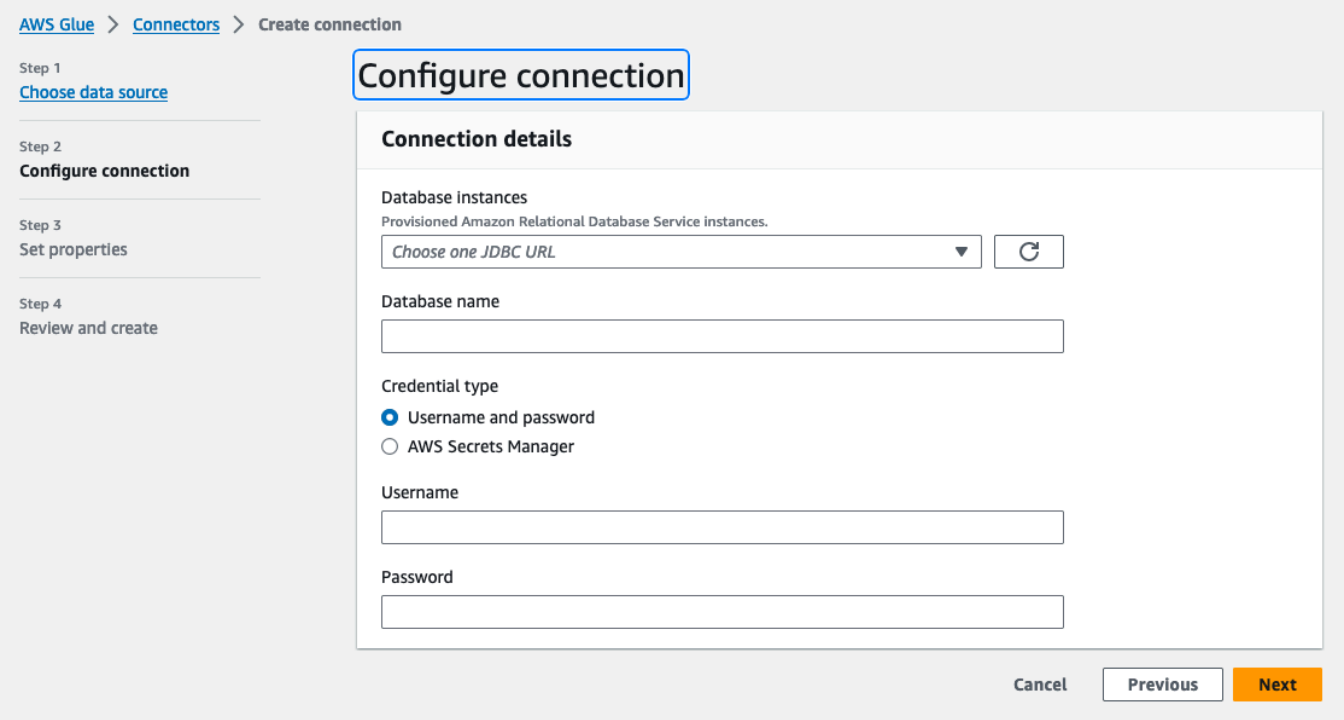


Depois de escolher a fonte de dados, escolha Avançar.

5. Configure a conexão na Etapa 2 do assistente.

   Insira os detalhes da conexão. Dependendo do tipo de conector selecionado, talvez seja necessário inserir informações adicionais de conexão. Elas podem incluir:

   - Detalhes da conexão: esses campos mudarão dependendo da fonte de dados à qual você está se conectando. Por exemplo, se você estiver se conectando a bancos de dados do Amazon DocumentDB, será necessário inserir o URL do Amazon DocumentDB. Se estiver se conectando ao Amazon Aurora, deverá escolher a instância do banco de dados e inserirá o nome do banco de dados. A seguir estão os detalhes da conexão necessários para o Amazon Aurora:



- Tipo de credencial: escolha entre Nome de usuário e senha ou AWS Secrets Manager. Insira as informações de autenticação solicitadas.
- Para conectores que usam JDBC, insira as informações necessárias para criar o URL do JDBC para o armazenamento de dados.
- Se você usar uma nuvem privada virtual (VPC), insira as informações de rede da VPC.

6. Defina as propriedades da conexão na etapa 3 do assistente. É possível adicionar uma descrição e tags como parte opcional dessa etapa. O nome é obrigatório e pré-preenchido com um valor padrão. Escolha Próximo.
7. Revise a fonte, os detalhes e as propriedades da conexão. Se precisar fazer alterações, escolha Editar para a etapa no assistente. Ao concluir, escolha Criar conexão.

   Escolha Criar conexão.

   Você retorna para a página Connectors (Conectores) e o banner informativo indica que a conexão que foi criada. Agora é possível usar a conexão em seus trabalhos AWS Glue Studio.

Criar uma conexão Kafka

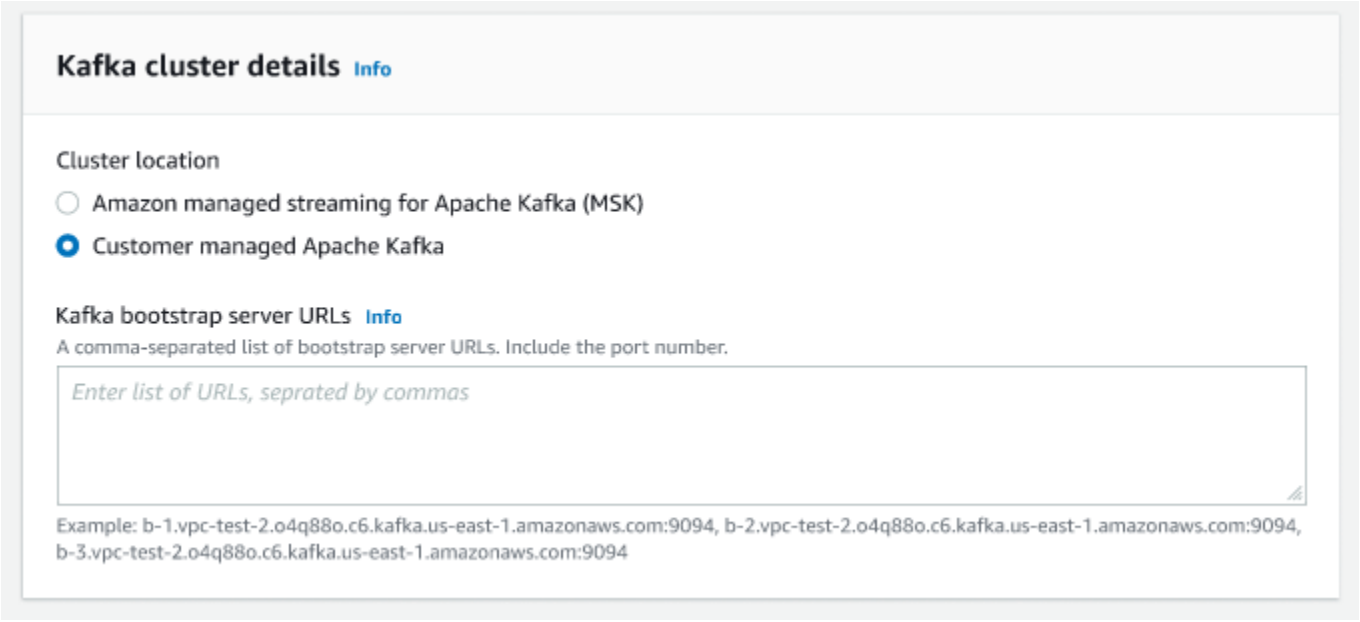
Ao criar uma conexão Kafka, selecionando Kafka no menu suspenso exibirá configurações adicionais para configurar:

- Detalhes do cluster do Kafka
- Autenticação
- Criptografia
- Opções de rede

Configurar detalhes do cluster do Kafka

1. Escolha o local do cluster. Você pode escolher entre um cluster Amazon Managed Streaming for Apache Kafka (MSK) ou um cluster Customer managed Apache Kafka. Para obter mais informações sobre o streaming gerenciado da Amazon para Apache Kafka, consulte Amazon Managed Streaming for Apache Kafka (MSK).

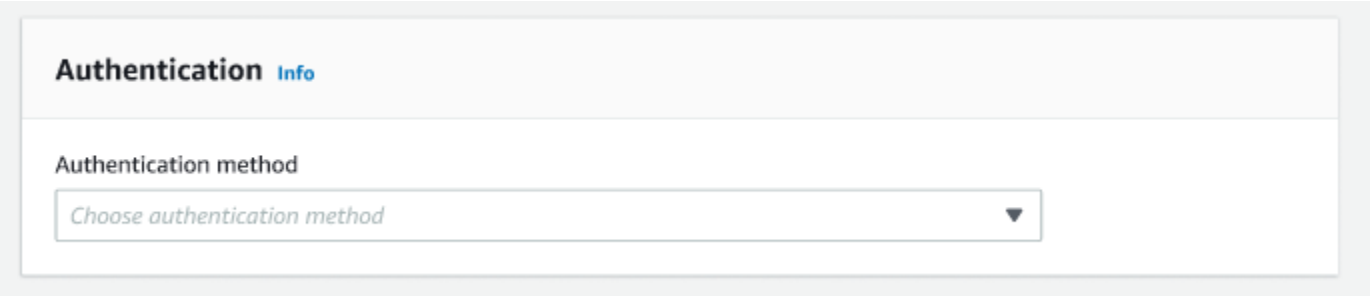
   > **Note**
   >
   > Amazon Managed Streaming for Apache Kafka suporta apenas métodos de autenticação TLS e SASL/SCRAM-SHA-512.



2. Insira os URLs para seus servidores de bootstrap Kafka. Você pode inserir mais de um separando cada servidor por uma vírgula. Inclua o número da porta no final da URL anexando `:<port number>`.

   Por exemplo: `b-1.vpc-test-2.034a88o.kafka-us-east-1.amazonaws.com:9094`

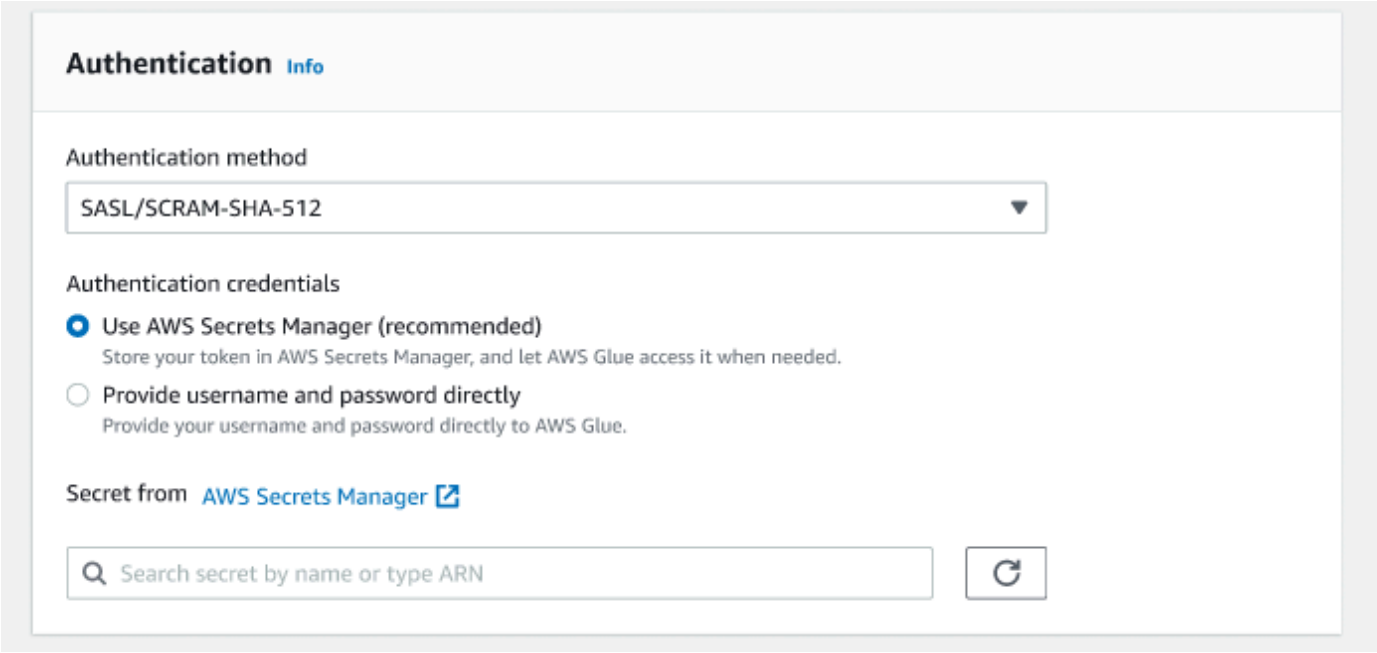
Selecione o método de autenticação



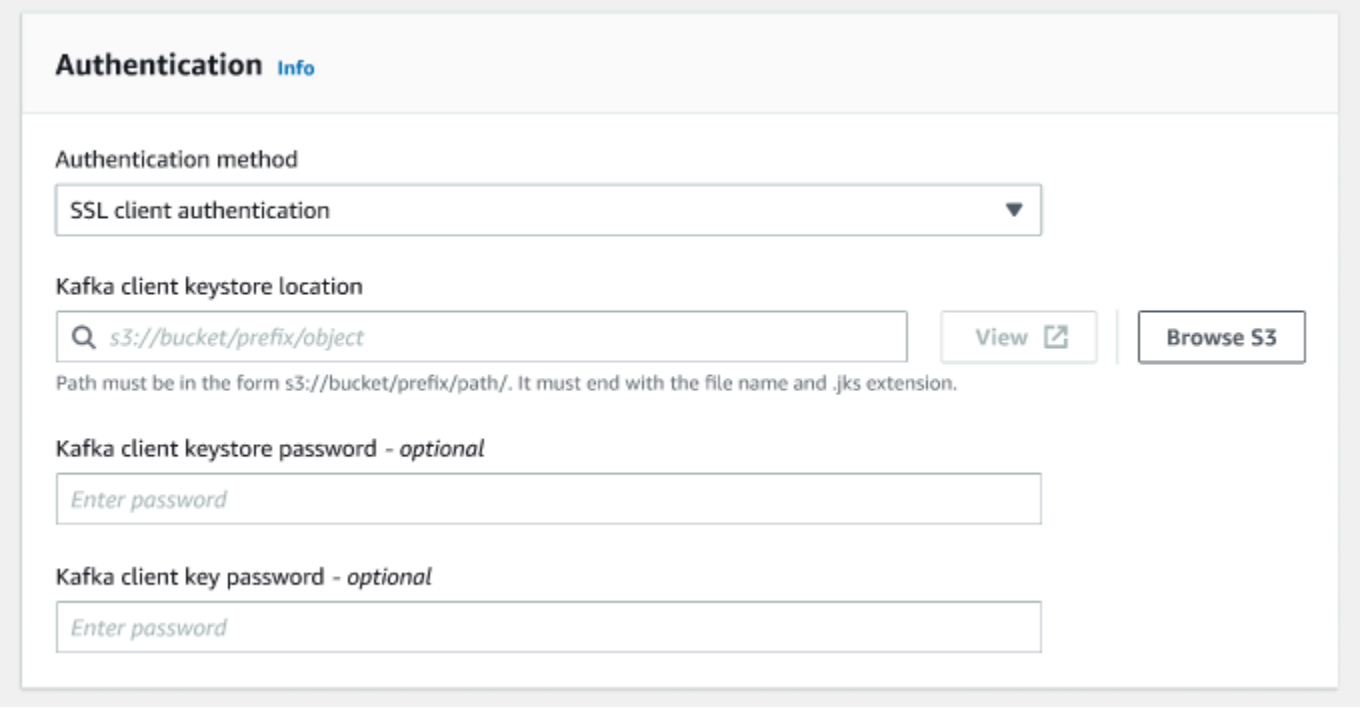
AWS Glue suporta a framework Simple Authentication and Security Layer (SASL) para autenticação. A estrutura SASL é compatível com vários mecanismos de autenticação e o AWS Glue oferece os protocolos SCRAM (nome de usuário e senha), GSSAPI (protocolo Kerberos) e PLAIN (nome de usuário e senha).

Ao escolher um método de autenticação no menu suspenso, os seguintes métodos de autenticação de cliente podem ser selecionados:

- Nenhum - Sem autenticação. Isso é útil se você criar uma conexão para fins de teste.
- SASL/SCRAM-SHA-512: escolha esse método de autenticação para especificar as credenciais de autenticação. Existem duas opções disponíveis:
  - Usar o AWS Secrets Manager (recomendado): se você selecionar essa opção, poderá armazenar suas credenciais no AWS Secrets Manager e deixar que o AWS Glue acesse as informações quando necessário. Especifique o segredo que armazena as credenciais de autenticação SSL ou SASL.



  - Forneça o nome de usuário e a senha diretamente.
- SASL/GSSAPI (Kerberos) - se você selecionar essa opção, poderá selecionar o local do arquivo keytab, arquivo krb5.conf e inserir o nome principal do Kerberos e o nome do serviço Kerberos. Os locais do arquivo keytab e do arquivo krb5.conf devem estar em um local do Amazon S3. Como o MSK ainda não oferece suporte a SASL/GSSAPI, essa opção está disponível apenas para clusters Apache Kafka gerenciados pelo cliente. Para obter mais informações, consulte Documentação do MIT Kerberos: Keytab.
- SASL/PLAIN: escolha esse método de autenticação para especificar as credenciais de autenticação. Existem duas opções disponíveis:
  - Usar o AWS Secrets Manager (recomendado): se você selecionar essa opção, poderá armazenar suas credenciais no AWS Secrets Manager e deixar que o AWS Glue acesse as informações quando necessário. Especifique o segredo que armazena as credenciais de autenticação SSL ou SASL.
  - Forneça o nome de usuário e a senha diretamente.
- Autenticação de cliente SSL - se você selecionar essa opção, poderá selecionar o local do keystore do cliente Kafka navegando no Amazon S3. Opcionalmente, você pode inserir a senha do keystore do cliente Kafka e a senha da chave do cliente Kafka.

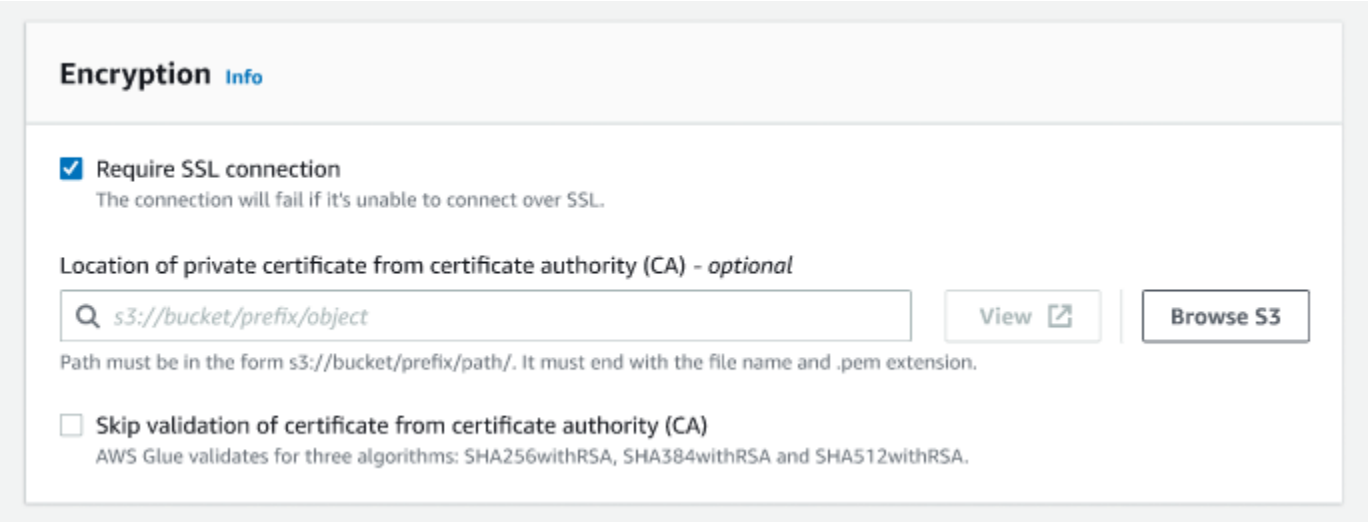


Definir as configurações de criptografia

1. Se a conexão Kafka exigir conexão SSL, marque a caixa de seleção para Exigir conexão SSL. Observe que a conexão falhará se não for possível se conectar por SSL. O SSL para criptografia pode ser usado com qualquer um dos métodos de autenticação (SASL/SCRAM-SHA-512, SASL/GSSAPI, SASL/PLAIN ou autenticação de cliente SSL) e é opcional.

   Se o método de autenticação estiver definido como Autenticação de cliente SSL, esta opção será selecionada automaticamente e será desativada para evitar alterações.
2. (Optional). Escolha o local do certificado privado da autoridade de certificação (CA). Observe que a localização da certificação deve estar em um local do S3. Selecione Browse (Navegar) para escolher o arquivo de um bucket S3 conectado. O caminho deve estar no formato `s3://bucket/prefix/filename.pem`. Deve terminar com o nome do arquivo e a extensão .pem.
3. Você pode optar por ignorar a validação do certificado de uma autoridade de certificação (CA). Escolha a caixa de seleção Ignorar a validação do certificado de uma autoridade de certificação (CA). Se esta caixa não estiver marcada, AWS Glue valida certificados para três algoritmos:

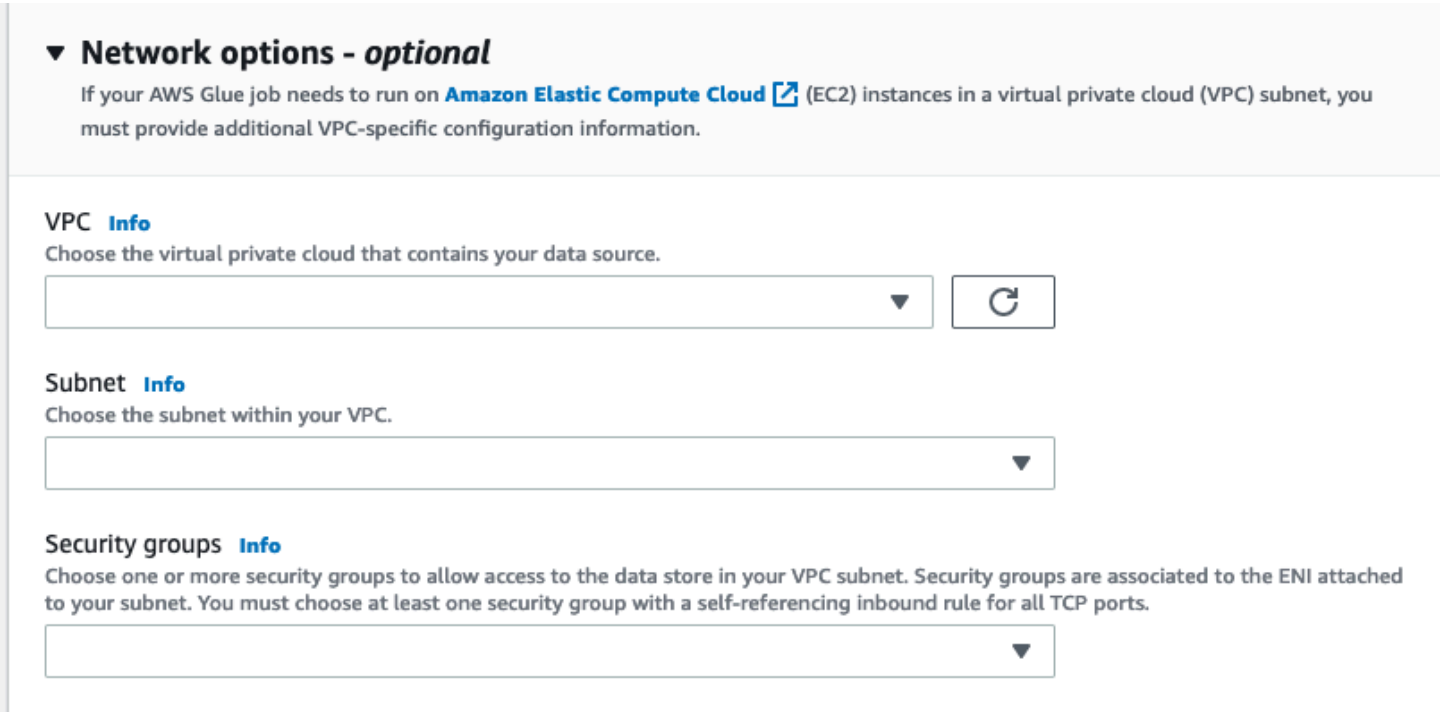
   - SHA256withRSA
   - SHA384withRSA
   - SHA512withRSA



(Opcional) Opções de rede

Veja a seguir as etapas opcionais para configurar VPC, sub-rede e grupos de segurança. Se seu trabalho AWS Glue precisar ser executado em instâncias do Amazon EC2 em uma sub-rede de nuvem privada virtual (VPC), você deverá fornecer informações adicionais de configuração específicas da VPC.

1. Escolha a VPC (nuvem privada virtual) que contém sua fonte de dados.
2. Escolha uma sub-rede com sua VPC.
3. Escolha um ou mais grupos de segurança para permitir o acesso ao datastore em sua sub-rede VPC. Grupos de segurança estão associados à ENI anexada à sua sub-rede. Você deve escolher pelo menos um grupo de segurança com uma regra de entrada de autorreferência para todas as portas TCP.



Criação de trabalhos com conectores personalizados

Você pode usar conectores e conexões para nós de origem e de destino de dados no AWS Glue Studio.

Tópicos

- Criar trabalhos que usam um conector para a origem dos dados
- Configurar propriedades de origem para nós que usam conectores
- Configurar propriedades de destino para nós que usam conectores

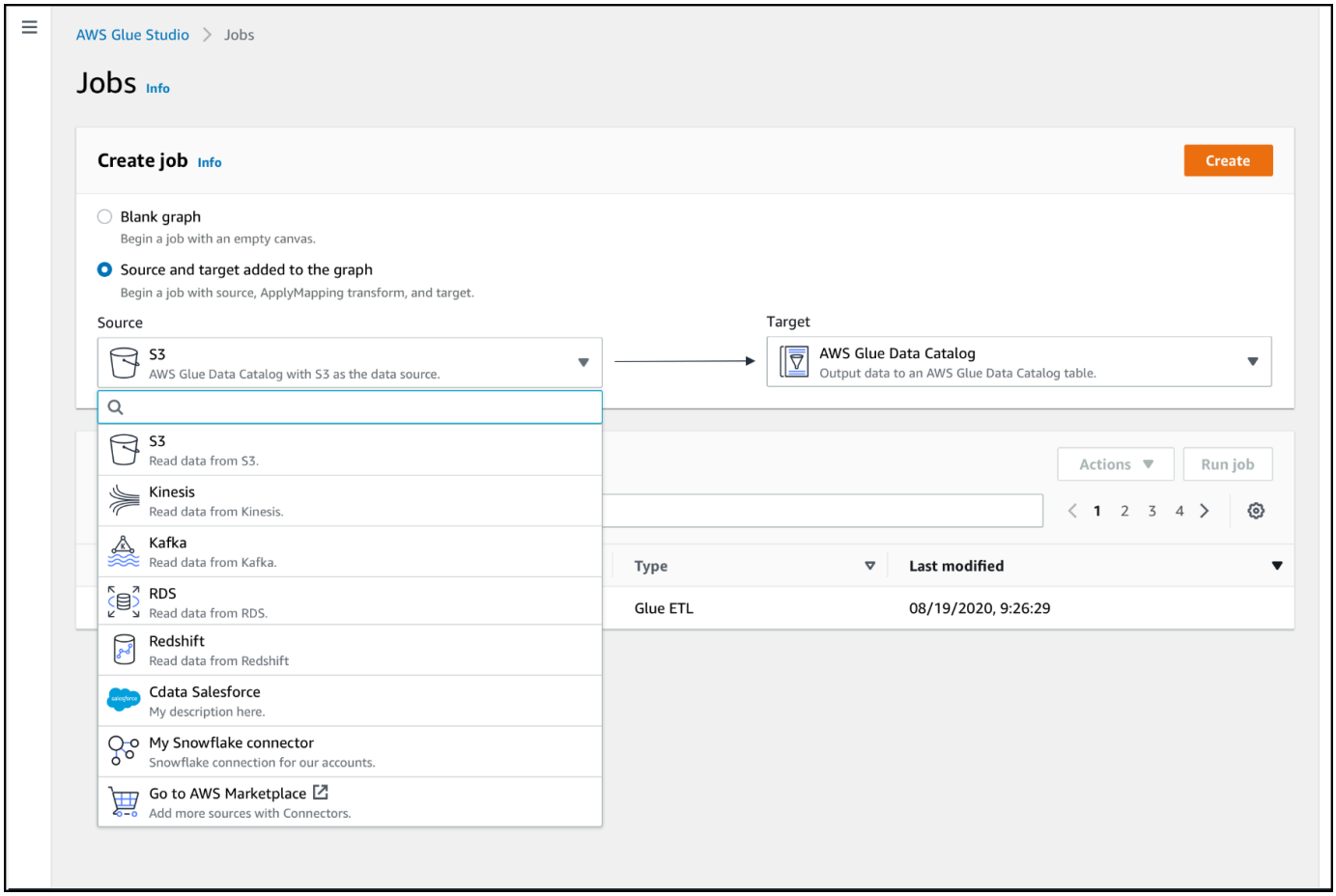
Criar trabalhos que usam um conector para a origem dos dados

Ao criar um novo trabalho, você pode escolher um conector para a origem e para os destinos dos dados.

Para criar um trabalho que use conectores para a origem ou destino dos dados

1. Faça login no AWS Management Console e abra o AWS Glue Studio console em https://console.aws.amazon.com/gluestudio/.
2. Na página Connectors (Conectores), na lista de recursos Your connections (Suas conexões), escolha a conexão que você deseja usar em seu trabalho e escolha Create job (Criar trabalho).

   Como alternativa, na página Jobs (Trabalhos) do AWS Glue Studio, em Create job (Criar trabalho), escolha Source and target added to the graph (Origem e destino adicionados ao gráfico). Na lista suspensa Source (Origem), escolha o conector personalizado que você deseja usar em seu trabalho. Também é possível escolher um conector para Target (Destino).



3. Escolha Create (Criar) para abrir o editor de trabalhos visual.

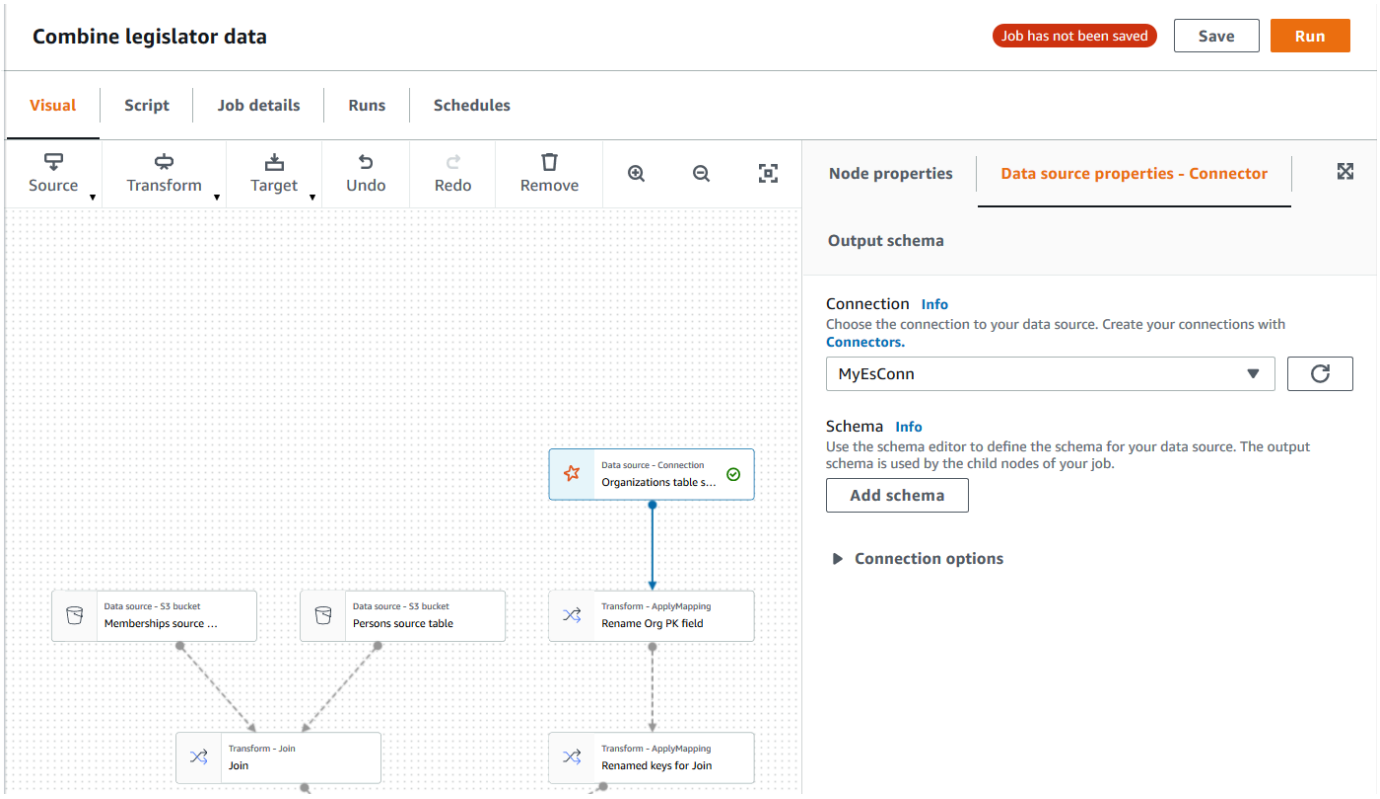
4. Configure o nó da origem dos dados, conforme descrito em Configurar propriedades de origem para nós que usam conectores.
5. Continue criando seu trabalho de ETL adicionando transformações, armazenamentos de dados adicionais e destinos de dados, conforme descrito em Visual ETL com AWS Glue Studio.
6. Personalize o ambiente de execução de trabalho configurando as propriedades do trabalho, conforme descrito em Modificar as propriedades do trabalho.
7. Salve o trabalho e o execute.

Configurar propriedades de origem para nós que usam conectores

Depois de criar um trabalho que usa um conector para a origem dos dados, o editor de trabalhos visual exibe um gráfico de trabalho com um nó de origem dos dados configurado para o conector. Você deve configurar as propriedades da origem dos dados para esse nó.

Para configurar as propriedades de um nó de origem dos dados que usa um conector

1. Escolha o nó da origem dos dados do conector no gráfico de trabalho ou adicione um novo nó e escolha o conector para Node type (Tipo de nó). Em seguida, no lado direito, no painel de detalhes do nó, escolha a guia Data source properties (Propriedades da origem dos dados), se ainda não estiver selecionada.



2. Na guia Data source properties (Propriedades da origem dos dados), escolha a conexão que você deseja usar para esse trabalho.

   Insira as informações adicionais necessárias para cada tipo de conexão:

   JDBC

   - Data source input type (Tipo de entrada da origem dos dados): opte por fornecer um nome de tabela ou uma consulta SQL como a origem dos dados. Dependendo da sua escolha, você precisará fornecer estas informações adicionais:
     - Table name (Nome da tabela): o nome da tabela na origem dos dados. Se a fonte de dados não usar o termo tabela, forneça o nome de uma estrutura de dados apropriada, conforme indicado pelas informações de uso do conector personalizado (que estão disponíveis em AWS Marketplace).
     - Filter predicate (Filtrar predicado): uma cláusula de condição a ser usada ao ler a origem dos dados, semelhante a uma cláusula `WHERE`, que é usada para recuperar um subconjunto dos dados.
     - Query code (Código de consulta): insira uma consulta SQL a ser usada para recuperar um conjunto de dados específico da origem dos dados. Um exemplo de uma consulta SQL básica é:

       ```
       SELECT column_list FROM
                                table_name WHERE where_clause
       ```

   - Schema (Esquema): como o AWS Glue Studio está usando informações armazenadas na conexão para acessar a origem dos dados em vez de recuperar informações de metadados de uma tabela do Data Catalog, você deve fornecer os metadados do esquema para a origem dos dados. Escolha Add schema (Adicionar esquema) para abrir o editor de esquemas.

     Para obter instruções sobre como usar o editor de esquemas, consulte Editar o esquema de um nó de transformação personalizada.
   - Partition column (Coluna da partição): (opcional) você pode optar por particionar as leituras de dados fornecendo valores para Partition column (Coluna da partição), Lower bound (Limite inferior), Upper bound (Limite superior) e Number of partitions (Número de partições).

Os valores de `lowerBound` e `upperBound` são usados para decidir o passo de partição, não para filtrar as linhas na tabela. Todas as linhas na tabela são particionadas e retornadas.

> **Note**
>
> O particionamento de colunas adiciona uma condição de particionamento extra à consulta usada para ler os dados. Ao usar uma consulta em vez de um nome de tabela, você deve validar se a consulta funciona com a condição de particionamento especificada. Por exemplo:
>
> - Se o seu formato de consulta for `"SELECT col1 FROM table1"`, teste a consulta anexando uma cláusula `WHERE` no final da consulta que usa a coluna de partição.
> - Se o seu formato de consulta for `"SELECT col1 FROM table1 WHERE col2=val"`, teste a consulta estendendo a cláusula `WHERE` com `AND` e uma expressão que usa a coluna de partição.

- Data type casting (Conversão de tipos de dados): se a origem dos dados usar tipos de dados que não estão disponíveis no JDBC, use essa seção para especificar como um tipo de dados da origem dos dados deve ser convertido em tipos de dados do JDBC. Você pode especificar até 50 conversões de tipos de dados diferentes. Todas as colunas na origem dos dados que usam o mesmo tipo de dados são convertidas da mesma maneira.

  Por exemplo, se você tiver três colunas na origem dos dados que usam o tipo de dados `Float`, e você indicar que o tipo de dados `Float` deve ser convertido para o tipo `String` do JDBC, então todas as três colunas que usam o tipo de dados `Float` são convertidas para o tipo de dados `String`.

- Job bookmark keys (Chaves de marcadores de trabalho): os marcadores de trabalho ajudam o AWS Glue a manter as informações de estado e a impedir o reprocessamento de dados antigos. Especifique uma ou mais colunas adicionais como chaves de marcadores. O AWS Glue Studio usa chaves de marcadores para rastrear dados que já foram processados durante uma execução anterior do trabalho de ETL. Todas as colunas que você usar para chaves de marcadores personalizadas devem ser estritamente monotônicas, aumentando ou diminuindo, mas lacunas são permitidas.

Se você inserir várias chaves de marcadores, elas serão combinadas para formar uma única chave composta. Uma chave de marcadores de trabalho composta não deve conter colunas duplicadas. Se você não especificar chaves de marcadores, por padrão, o AWS Glue Studio usará a chave primária como chave de marcadores, desde que ela esteja aumentando ou diminuindo sequencialmente (sem lacunas). Se a tabela não tiver uma chave primária, mas a propriedade de marcador de trabalho estiver habilitada, você deverá fornecer chaves de marcadores de trabalho personalizadas. Caso contrário, a pesquisa de chaves primárias a serem usadas como padrão falhará, assim como a execução do trabalho.

- Job bookmark keys sorting order (Ordem de classificação de chaves de marcadores de trabalhos): escolha se as chaves-valor aumentam ou diminuem sequencialmente.

Spark

- Schema (Esquema): como o AWS Glue Studio está usando informações armazenadas na conexão para acessar a origem dos dados em vez de recuperar informações de metadados de uma tabela do Data Catalog, você deve fornecer os metadados do esquema para a origem dos dados. Escolha Add schema (Adicionar esquema) para abrir o editor de esquemas.

  Para obter instruções sobre como usar o editor de esquemas, consulte Editar o esquema de um nó de transformação personalizada.

- Connection options (Opções de conexão): insira pares de chave-valor adicionais, conforme necessário, para fornecer informações ou opções de conexão adicionais. Por exemplo, você pode inserir um nome de banco de dados, nome de tabela, nome de usuário e senha.

  Por exemplo, para OpenSearch, você insere os seguintes pares de valores-chave, conforme descrito em: the section called “ Tutorial: Usar o 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 um exemplo das opções mínimas de conexão a serem usadas, consulte o exemplo de script de teste MinimalSparkConnectorTest.scala on GitHub, que mostra as opções de conexão que você normalmente forneceria em uma conexão.

Athena

- Table name (Nome da tabela): o nome da tabela na origem dos dados. Se você estiver usando um conector para ler os CloudWatch registros do Athena, insira o nome da tabela. `all_log_streams`
- Athena schema name (Nome do esquema do Athena): escolha o esquema na origem dos dados do Athena que corresponde ao banco de dados que contém a tabela. Se você estiver usando um conector para ler os CloudWatch registros do Athena, insira um nome de esquema semelhante a. `/aws/glue/`*`name`*
- Schema (Esquema): como o AWS Glue Studio está usando informações armazenadas na conexão para acessar a origem dos dados em vez de recuperar informações de metadados de uma tabela do Data Catalog, você deve fornecer os metadados do esquema para a origem dos dados. Escolha Add schema (Adicionar esquema) para abrir o editor de esquemas.

  Para obter instruções sobre como usar o editor de esquemas, consulte Editar o esquema de um nó de transformação personalizada.
- Additional connection options (Opções de conexão adicionais): insira pares de chave-valor adicionais, conforme necessário, para fornecer informações ou opções de conexão adicionais.

Para ver um exemplo, consulte o `README.md` arquivo em https://github.com/aws-samples/aws-glue-samples /tree/master/ GlueCustomConnectors /development/Athena. Nas etapas deste documento, o código de exemplo mostra as opções de conexão mínimas necessárias, que são `tableName`, `schemaName` e `className`. O código de exemplo especifica essas opções como parte da variável `optionsMap`, mas você pode especificá-las para sua conexão e, em seguida, usar a conexão.

3. (Opcional) depois de fornecer as informações necessárias, você pode exibir o esquema de dados resultante para sua origem dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. O esquema exibido nessa guia é usado por todos os nós filhos adicionados ao gráfico de trabalho.

4. (Opcional) depois de configurar as propriedades do nó e da origem dos dados, você poderá previsualizar o conjunto de dados de sua origem dos dados escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Configurar propriedades de destino para nós que usam conectores

Se você usar um conector para o tipo de destino de dados, deverá configurar as propriedades do nó de destino de dados.

Para configurar as propriedades de um nó de destino de dados que usa um conector

1. Escolha o nó de destino de dados do conector no gráfico de trabalho. Em seguida, no lado direito, no painel de detalhes do nó, escolha a guia Data target properties (Propriedades de destino de dados), se ainda não estiver selecionada.
2. Na guia Data target properties (Propriedades de destino de dados), escolha a conexão a ser usada para gravar no destino.

   Insira as informações adicionais necessárias para cada tipo de conexão:

   JDBC

   - Connection (Conexão): escolha a conexão a ser usada com o conector. Para obter informações sobre como criar uma conexão, consulte Criar conexões para conectores.
   - Table name (Nome da tabela): o nome da tabela no destino dos dados. Se o destino de dados não usar o termo tabela, forneça o nome de uma estrutura de dados apropriada, conforme indicado pelas informações de uso do conector personalizado (que estão disponíveis em AWS Marketplace).
   - Batch size (Tamanho do lote; opcional): informe o número de linhas ou registros a serem inseridos na tabela de destino em uma única operação. O valor padrão é 1.000 linhas.

   Spark

   - Connection (Conexão): escolha a conexão a ser usada com o conector. Se você não criou uma conexão anteriormente, escolha Create connection (Criar conexão) para criar uma.

Para obter informações sobre como criar uma conexão, consulte Criar conexões para conectores.

- Connection options (Opções de conexão): insira pares de chave-valor adicionais, conforme necessário, para fornecer informações ou opções de conexão adicionais. Você pode inserir um nome de banco de dados, um nome de tabela, um nome de usuário e senha.

  Por exemplo, para OpenSearch, você insere os seguintes pares de valores-chave, conforme descrito em: the section called " Tutorial: Usar o 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 um exemplo das opções mínimas de conexão a serem usadas, consulte o exemplo de script de teste MinimalSparkConnectorTest.scala on GitHub, que mostra as opções de conexão que você normalmente forneceria em uma conexão.

3. Depois de fornecer as informações necessárias, você pode exibir o esquema de dados resultante para sua origem dos dados escolhendo a guia Output schema (Esquema de saída), no painel de detalhes do nó.

Gerenciar conectores e conexões

Você pode usar a página Conexões no AWS Glue para gerenciar seus conectores e conexões.

Tópicos

- Visualizar detalhes do conector e da conexão
- Editar conectores e conexões
- Excluir conectores e conexões
- Cancelar a assinatura de um conector

Visualizar detalhes do conector e da conexão

Você pode exibir informações resumidas sobre seus conectores e conexões nas tabelas de recursos Your connectors (Seus conectores) e Your connections (Suas conexões) na página Connectors (Conectores). Para visualizar as informações detalhadas, execute as etapas a seguir.

Para exibir detalhes do conector ou da conexão

1. No console do AWS Glue Studio, escolha Connectors (Conectores), no painel de navegação do console.
2. Escolha o conector ou conexão cujas informações detalhadas você deseja visualizar.
3. Escolha Actions (Ações) e, em seguida, escolha View details (Visualizar os detalhes), para abrir a página de detalhes desse conector ou conexão.
4. Na página de detalhes, você pode optar por Edit (Editar) ou Delete (Excluir) o conector ou a conexão.

   - Para conectores, você pode escolher Create connection (Criar conexão) para criar uma nova conexão que use o conector.
   - Para conexões, você pode escolher Create job (Criar trabalho) para criar um novo trabalho que use a conexão.

Editar conectores e conexões

Você usa a página Connectors (Conectores) para alterar as informações armazenadas em seus conectores e conexões.

Para modificar um conector ou conexão

1. No console do AWS Glue Studio, escolha Connectors (Conectores), no painel de navegação do console.
2. Escolha o conector ou conexão que você deseja alterar.
3. Escolha Ações e, em seguida, escolha Editar.

   Você também pode escolher View details (Visualizar os detalhes) e, na página de detalhes do conector ou da conexão, escolher Edit (Editar).
4. Na página Edit connector (Editar conector) ou Edit connection (Editar conexão), atualize as informações e escolha Save (Salvar).

Excluir conectores e conexões

Você pode usar a página Connectors (Conectores) para excluir conectores e conexões. Se você excluir um conector, todas as conexões que foram criadas para ele também devem ser excluídas.

Para remover conectores do AWS Glue Studio

1. No console do AWS Glue Studio, escolha Connectors (Conectores), no painel de navegação do console.
2. Escolha o conector ou conexão que você deseja excluir.
3. Escolha Ações e, em seguida, escolha Excluir.

   Você também pode escolher View details (Visualizar os detalhes) e, na página de detalhes do conector ou da conexão, escolher Delete (Excluir).
4. Confirme que você deseja remover o conector ou a conexão digitando **`Delete`** e escolha Delete (Excluir).

   Se você excluir um conector, todas as conexões que foram criadas para ele também devem ser excluídas.

Quaisquer trabalhos que utilizem uma conexão excluída deixarão de funcionar. Você pode editar os trabalhos para usar um datastore diferente ou removê-los. Para obter informações sobre como excluir um trabalho, consulte Excluir trabalhos.

Se você excluir um conector, isso não cancelará a assinatura do conector no AWS Marketplace. Para remover a assinatura de um conector excluído, siga as instruções em Cancelar a assinatura de um conector.

Cancelar a assinatura de um conector

Depois de excluir as conexões e o conector deAWS Glue Studio, você pode cancelar sua assinatura AWS Marketplace se não precisar mais do conector.

> **Note**
>
> Se você cancelar a assinatura de um conector, isso não removerá o conector ou a conexão da sua conta. Quaisquer trabalhos que usem o conector e as conexões relacionadas não poderão mais usá-lo e falharão.

Antes de cancelar a assinatura ou assinar novamente um conector de AWS Marketplace, você deve excluir as conexões existentes e os conectores associados a esse produto. AWS Marketplace

Para cancelar a assinatura de um conector em AWS Marketplace

1. Faça login no AWS Marketplace console em https://console.aws.amazon.com/marketplace.
2. Escolha Manage subscriptions (Gerenciar assinaturas).
3. Na página Manage subscriptions (Gerenciar assinaturas), escolha Manage (Gerenciar), ao lado da assinatura do conector que você deseja cancelar.
4. Escolha Actions (Ações) e escolha Cancel subscription (Cancelar assinatura).
5. Marque a caixa de seleção para confirmar que as instâncias em execução são cobradas em sua conta e escolha Yes, cancel subscription (Sim, cancelar assinatura).

Desenvolvimento de conectores personalizados

Você pode escrever código para ler ou gravar dados no seu datastore e formatar esses dados para usá-los com trabalhos do AWS Glue Studio. Você pode criar conectores para armazenamentos de dados do Spark, Athena e JDBC. O código de amostra publicado em GitHub fornece uma visão geral das interfaces básicas que você precisa implementar.

Você precisará de um ambiente de desenvolvimento local para criar seu código de conector. Você pode usar qualquer IDE ou até mesmo um simples editor de linha de comando para escrever seu conector. Exemplos de ambientes de desenvolvimento incluem:

- Um ambiente Scala local com uma biblioteca local de ETL Maven do AWS Glue, conforme descrito em Desenvolver localmente com o Scala no Guia do desenvolvedor do AWS Glue .
- IntelliJ IDE, baixando o IDE de https://www.jetbrains.com/idea/.

Tópicos

- Desenvolvimento de conectores do Spark
- Desenvolvimento de conectores do Athena
- Desenvolvimento de conectores do JDBC
- Exemplos de uso de conectores personalizados com o AWS Glue Studio

- Desenvolvendo AWS Glue conectores para AWS Marketplace

Desenvolvimento de conectores do Spark

Você pode criar um conector Spark com a DataSource API Spark V2 (Spark 2.4) para ler dados.

Para criar um conector personalizado do Spark

Siga as etapas na biblioteca de AWS Glue GitHub amostras para desenvolver conectores Spark, localizada em https://github.com/aws-samples/ aws-glue-samples /tree/master/ GlueCustomConnectors /development/spark/README.md.

Desenvolvimento de conectores do Athena

Você pode criar um conector do Athena para ser usado pelo AWS Glue e pelo AWS Glue Studio para consultar uma origem dos dados personalizada.

Para criar um conector personalizado do Athena

Siga as etapas na biblioteca de AWS Glue GitHub amostras para desenvolver conectores Athena, localizada em https://github.com/aws-samples/ aws-glue-samples /tree/master/ /development/Athena. GlueCustomConnectors

Desenvolvimento de conectores do JDBC

Você pode criar um conector que usa o JDBC para acessar seus armazenamentos de dados.

Para criar um conector do JDBC personalizado

1. Instale as bibliotecas do runtime Spark do AWS Glue em seu ambiente de desenvolvimento local. Consulte as instruções na biblioteca de AWS Glue GitHub amostras em https://github.com/aws-samples/ aws-glue-samples /tree/master/ /development/ GlueCustomConnectors / README.md. GlueSparkRuntime
2. Implante o driver do JDBC responsável por recuperar dados da origem dos dados. Consulte a documentação do Java para o Java SE 8.

   Crie um ponto de entrada dentro do seu código que o AWS Glue Studio usará para localizar seu conector. O campo Class name (Nome da classe) deve ser o caminho completo do driver do JDBC.
3. Use a API `GlueContext` para ler dados com o conector. Os usuários podem adicionar mais opções de entrada no console do AWS Glue Studio para configurar a conexão com a origem

dos dados, se necessário. Para ver um exemplo de código que mostra como ler e gravar em um banco de dados JDBC com um conector JDBC personalizado, consulte Valores personalizados e ConnectionType. AWS Marketplace

Exemplos de uso de conectores personalizados com o AWS Glue Studio

Você pode consultar os seguintes blogs para obter exemplos de uso de conectores personalizados:

- Desenvolvimento, teste e implantação de conectores personalizados para seus armazenamentos de dados com o AWS Glue
- Apache Hudi: Escrever tabelas Apache Hudi usando o AWS Glue Conector Personalizado
- Google BigQuery: migrando dados do Google BigQuery para o Amazon S3 AWS Glue usando conectores personalizados
- Snowflake (JDBC): Realizar transformações de dados usando o Snowflake e AWS Glue
- SingleStore: Construindo ETL rápido usando SingleStore e AWS Glue
- Salesforce: Insirir dados do Salesforce no Amazon S3 usando o conector personalizado CData JDBC com AWS Glue -
- MongoDB: Construir AWS Glue trabalhos Spark ETL usando o Amazon DocumentDB (com compatibilidade MongoDB) e o MongoDB
- Amazon Relational Database Service (Amazon RDS): Crie trabalhos de ETL no AWS Glue Spark trazendo seus próprios drivers JDBC para o Amazon RDS
- MySQL (JDBC): https://github.com/aws-samples/ /blob/master/ /development/spark/ SQL.scala aws-glue-samples GlueCustomConnectors SparkConnectorMy

Desenvolvendo AWS Glue conectores para AWS Marketplace

Como AWS parceiro, você pode criar conectores personalizados e enviá-los para vender AWS Marketplace aos AWS Glue clientes.

O processo para desenvolver o código do conector é o mesmo que para conectores personalizados, mas o processo de carregamento e verificação do código do conector é mais detalhado. Consulte as instruções em Criação de conectores AWS Marketplace no GitHub site.

Restrições de uso de conectores e conexões no AWS Glue Studio

Ao usar conectores personalizados ou conectores da AWS Marketplace, observe as seguintes restrições:

- A API testConnection não é suportada com conexões criadas para conectores personalizados.
- A criptografia de senha de conexão do Data Catalog não é suportada com conectores personalizados.
- Você não pode usar marcadores de trabalho se especificar um predicado de filtro para um nó de fonte de dados que usa um conector JDBC.
- A criação de uma conexão com o Marketplace não é suportada fora da interface AWS Glue Studio do usuário.

Conectar a fontes de dados usando trabalhos do Visual ETL

Ao criar um novo trabalho, você pode usar conexões para conectar aos dados ao editar trabalhos visuais de ETL no AWS Glue. Você pode fazer isso adicionando nós de origem que usam conectores para ler dados e nós de destino para especificar o local para gravação dos dados.

Tópicos

Modificar as propriedades de um nó da fonte dos dados

Para especificar as propriedades da origem dos dados, escolha primeiro um nó de origem dos dados no diagrama de trabalho. Em seguida, no lado direito do painel de detalhes do nó, configure as propriedades do nó.

Para modificar as propriedades de um nó de origem dos dados

1. Vá para o editor visual para um trabalho novo ou salvo.
2. Escolha um nó de origem dos dados no diagrama de trabalho.
3. Selecione a guia Node properties (Propriedades do nó) no painel de detalhes do nó e insira as seguintes informações:

- Name (Nome): (opcional) insira um nome a ser associado ao nó no diagrama de trabalho. Esse nome deve ser exclusivo entre todos os nós para esse trabalho.
- Node type (Tipo de nó): o tipo de nó determina a ação que é executada pelo nó. Na lista de opções de Node type (Tipo de nó), escolha um dos valores listados no cabeçalho Data source (Origem dos dados).

4. Configure as informações de Data source properties (Propriedades da origem dos dados). Para obter mais informações, consulte as seções a seguir:

   - Usar as tabelas do Data Catalog para a origem dos dados
   - Usar um conector para a origem dos dados
   - Usar arquivos no Amazon S3 para a origem dos dados
   - Usar uma fonte de dados de transmissão

5. (Opcional) depois de configurar as propriedades do nó e da origem dos dados, você pode exibir o esquema de sua origem dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.
6. (Opcional) depois de configurar as propriedades do nó e da origem dos dados, você poderá previsualizar o conjunto de dados de sua origem dos dados escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Usar as tabelas do Data Catalog para a origem dos dados

Para todas as origens dos dados, exceto o Amazon S3 e os conectores, uma tabela deve existir noAWS Glue Data Catalog para o tipo de origem que você escolher. O AWS Glue não cria a tabela do Data Catalog.

Para configurar um nó de origem dos dados com base em uma tabela do Data Catalog

1. Vá para o editor visual para um trabalho novo ou salvo.
2. Escolha um nó de origem dos dados no diagrama de trabalho.

3. Escolha a guia Data source properties (Propriedades da origem dos dados) e insira as seguintes informações:

   - S3 source type (Tipo de origem do S3): (somente para origens de dados do Amazon S3) escolha a opção Select a Catalog table (Selecionar uma tabela do catálogo) para usar uma tabela do AWS Glue Data Catalog existente.
   - Database (Banco de dados): escolha o banco de dados no Data Catalog que contém a tabela de origem que você deseja usar para esse trabalho. É possível usar o campo de pesquisa para procurar um banco de dados pelo nome.
   - Table (Tabela): escolha a tabela associada aos dados de origem na lista. Essa tabela já deve existir no AWS Glue Data Catalog. É possível usar o campo de pesquisa para procurar uma tabela pelo nome.
   - Partition predicate (Predicado de partição): (somente para origens de dados do Amazon S3) insira uma expressão booleana baseada no Spark SQL que inclua apenas as colunas de particionamento. Por exemplo: `"(year=='2020' and month=='04')"`
   - Temporary directory (Diretório temporário): (somente para origens de dados do Amazon RedShift) insira um caminho para o local de um diretório funcional no Amazon S3, onde seu trabalho de ETL possa gravar resultados intermediários temporários.
   - Role associated with the cluster (Função associada ao cluster): (somente para origens de dados do Amazon RedShift) insira uma função para o seu trabalho de ETL usar que contenha permissões para clusters do Amazon Redshift. Para ter mais informações, consulte the section called “Permissões de origem e destino dos dados”.

Usar um conector para a origem dos dados

Se você selecionar um conector para Node type (Tipo de nó), siga as instruções em Criação de trabalhos com conectores personalizados para concluir a configuração das propriedades da origem dos dados.

Usar arquivos no Amazon S3 para a origem dos dados

Se você escolher o Amazon S3 como sua origem dos dados, poderá escolher:

- Um banco de dados e tabela do Data Catalog.
- Um bucket, pasta ou arquivo no Amazon S3.

Se você usar um bucket do Amazon S3 como origem dos dados, o AWS Glue detectará o esquema dos dados no local especificado a partir de um dos arquivos ou usando o arquivo especificado como um arquivo de exemplo. A detecção de esquema ocorre quando você usa o botão Infer schema (Inferir esquema). Se você alterar o local do Amazon S3 ou o arquivo de exemplo, escolha Infer schema (Inferir esquema) novamente para realizar a detecção de esquema usando as novas informações.

Para configurar um nó de origem dos dados que lê diretamente de arquivos no Amazon S3

1. Vá para o editor visual para um trabalho novo ou salvo.
2. Escolha um nó de origem dos dados no diagrama de trabalho para uma origem do Amazon S3.
3. Escolha a guia Data source properties (Propriedades da origem dos dados) e insira as seguintes informações:

   - S3 source type (Tipo de origem do S3): (somente para origens de dados do Amazon S3) escolha a opção S3 location (Localização do S3).
   - S3 URL (URL do S3): insira o caminho para o bucket, pasta ou arquivo do Amazon S3 que contém os dados do seu trabalho. Você pode escolher Browse S3 (Procurar no S3) para selecionar o caminho a partir dos locais disponíveis para sua conta.
   - Recursive (Recursivo): escolha essa opção se desejar que o AWS Glue leia dados de arquivos em pastas filhas no local do S3.

     Se as pastas filhas contiverem dados particionados, o AWS Glue não adicionará ao Data Catalog nenhuma informação de partição especificada nos nomes de pastas. Por exemplo, considere as seguintes pastas no Amazon S3:

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

     Se você escolher Recursive (Recursivo) e selecionar a pasta `sales` como a sua localização do S3, o AWS Glue lerá os dados em todas as pastas filhas, mas não criará partições para o ano, mês ou dia.
   - Data format (Formato de dados): escolha o formato no qual os dados são armazenados. Você pode escolher JSON, CSV ou Parquet. O valor selecionado informa ao trabalho do AWS Glue como ler os dados do arquivo de origem.

> **Note**
>
> Se você não selecionar o formato correto para os seus dados, O AWS Glue poderá inferir o esquema corretamente, mas o trabalho não poderá analisar corretamente os dados do arquivo de origem.

Você pode inserir opções de configuração adicionais, dependendo do formato escolhido.

- JSON (JavaScript Object Notation)
  - JsonPath (Caminho Json): insira um caminho JSON que aponta para um objeto usado para definir um esquema de tabela. Expressões de caminho JSON sempre se referem a uma estrutura JSON da mesma forma como expressões XPath são usadas em combinação com um documento XML. O “objeto de membro raiz” no caminho JSON é sempre referido como $, mesmo que seja um objeto ou matriz. O caminho JSON pode ser escrito na notação de pontos ou de colchetes.

    Para obter mais informações sobre operadores do caminho JSON, consulte JsonPath no site do GitHub.
  - Records in source files can span multiple lines (Registros em arquivos de origem podem abranger várias linhas): escolha essa opção se um único registro puder abranger várias linhas no arquivo CSV.
- CSV (comma-separated values)
  - Delimiter (Delimitador): insira um caractere para indicar o que separa cada entrada de coluna na linha, por exemplo, `;` ou `,`.
  - Escape character (Caractere de escape): insira um caractere usado como caractere de escape. Ele indica que o caractere que segue imediatamente o caractere de escape deve ser entendido literalmente e não deve ser interpretado como um delimitador.
  - Quote character (Caractere de aspas): insira o caractere usado para agrupar strings separadas em um único valor. Por exemplo, você escolheria aspas duplas (") se tiver valores como `"This is a single value"` em seu arquivo CSV.
  - Records in source files can span multiple lines (Registros em arquivos de origem podem abranger várias linhas): escolha essa opção se um único registro puder abranger várias linhas no arquivo CSV.

- First line of source file contains column headers (A primeira linha do arquivo de origem contém cabeçalhos de coluna): escolha essa opção se a primeira linha do arquivo CSV contiver cabeçalhos de coluna em vez de dados.

- Parquet (Apache Parquet columnar storage)

  Não há configurações adicionais a configurar para dados armazenados no formato Parquet.

- Partition predicate (Predicato de partição): para particionar os dados que são lidos a partir da origem dos dados, introduza uma expressão booliana baseada no Spark SQL que inclua apenas as colunas de particionamento. Por exemplo: `"(year=='2020' and month=='04')"`
- Advanced options (Opções avançadas): expanda essa seção se desejar que AWS Glue detecte o esquema de seus dados com base em um arquivo específico.
  - Schema inference (Inferência do esquema): escolha a opção Choose a sample file from S3(Escolher um arquivo de exemplo no S3), se desejar usar um arquivo específico em vez de deixar que o AWS Glue escolha um por você.
  - Auto-sampled file (Arquivo amostrado automaticamente): insira o caminho para o arquivo no Amazon S3 a ser usado para inferir o esquema.

  Se estiver editando um nó de origem dos dados e alterando o arquivo de exemplo selecionado, escolha Reload schema (Recarregar esquema) para detectar o esquema usando o novo arquivo de exemplo.

4. Selecione o botão Infer schema (Inferir esquema) para detectar o esquema dos arquivos de origem no Amazon S3. Se você alterar o local do Amazon S3 ou o arquivo de exemplo, deve escolher Infer schema (Inferir esquema) novamente para realizar a detecção do esquema usando as novas informações.

Usar uma fonte de dados de transmissão

É possível criar trabalhos de extração, transformação e carregamento (ETL) de transmissão que sejam executados continuamente e que consumam dados de fontes de transmissão, como o Amazon Kinesis Data Streams, Apache Kafka e Amazon Managed Streaming for Apache Kafka (Amazon MSK).

Para configurar propriedades de uma fonte de dados de transmissão

1. Vá para o editor visual de um trabalho novo ou salvo.

2. Escolha um nó de origem dos dados no gráfico para fluxos de dados do Kafka ou do Kinesis.
3. Escolha a guia Data source properties (Propriedades da origem dos dados) e insira as seguintes informações:

   Kinesis

   - Kinesis source type (Tipo de origem do Kinesis): escolha a opção Stream details (Detalhes do fluxo) para usar o acesso direto à fonte de transmissão ou escolha Data Catalog Table(Tabela do Data Catalog) para usar as informações nele armazenadas.

     Se você escolher Stream details (Detalhes do fluxo), especifique as informações adicionais a seguir.

     - Local do fluxo de dados: escolha se o fluxo está associado ao usuário atual ou a outro usuário.
     - Region (Região): escolha a Região da AWS onde o fluxo existe. Essas informações são usadas para construir o ARN para acesso ao fluxo de dados.
     - Stream ARN (ARN do fluxo): o nome do recurso da Amazon (ARN) do fluxos de dados do Kinesis. Se o fluxo estiver localizado na conta atual, você poderá escolher o nome do fluxo na lista suspensa. É possível usar o campo de pesquisa para procurar um fluxo de dados por seu nome ou ARN.
     - Data format (Formato de dados): escolha na lista o formato usado pelo fluxo de dados.

       O AWS Glue detecta automaticamente o esquema dos dados da transmissão.

     Se você escolher Data Catalog table (Tabela do Data Catalog), especifique as informações adicionais a seguir.

     - Database (Banco de dados): (opcional) escolha o banco de dados no AWS Glue Data Catalog que contém a tabela associada à sua fonte de dados de transmissão. É possível usar o campo de pesquisa para procurar um banco de dados pelo nome.
     - Table (Tabela): (opcional) escolha a tabela associada aos dados de origem na lista. Essa tabela já deve existir no AWS Glue Data Catalog. É possível usar o campo de pesquisa para procurar uma tabela pelo nome.
     - Detect schema (Detectar esquema): escolha essa opção para que o AWS Glue detecte o esquema dos dados da transmissão, em vez de usar as informações do esquema em uma tabela do Data Catalog. Essa opção é habilitada automaticamente quando a opção Stream details (Detalhes do fluxo) é escolhida.

- Starting position (Posição inicial): por padrão, o trabalho de ETL usa a opção Earliest (Mais antiga), o que significa que ele lê dados começando com o registro mais antigo disponível no fluxo. Em vez disso, você pode escolher Latest (Mais recente), o que indica que o trabalho de ETL deve começar a leitura logo após o registro mais recente no fluxo.
- Window size (Tamanho da janela): por padrão, o trabalho de ETL processa e grava dados em janelas de 100 segundos. Isso permite que os dados sejam processados de forma eficiente e que as agregações sejam realizadas em dados que chegam mais tarde do que o esperado. É possível modificar esse tamanho da janela para aumentar a pontualidade ou a precisão da agregação.

  Os trabalhos de transmissão do AWS Glue usam pontos de verificação em vez de marcadores de trabalho para rastrear os dados que foram lidos.
- Connection options (Opções de conexão): expanda essa seção para adicionar pares de chave-valor a fim de especificar opções de conexão adicionais. Para obter informações sobre quais opções você pode especificar aqui, consulte "connectionType": "kinesis" no Guia do desenvolvedor do AWS Glue.

Kafka

- Apache Kafka source (Origem do Apache Kafka): escolha a opção Stream details (Detalhes do fluxo) para usar o acesso direto à fonte de transmissão ou escolha Data Catalog Table(Tabela do Data Catalog) para usar as informações nele armazenadas.

  Se você escolher Data Catalog table (Tabela do Data Catalog), especifique as informações adicionais a seguir.
  - Database (Banco de dados): (opcional) escolha o banco de dados no AWS Glue Data Catalog que contém a tabela associada à sua fonte de dados de transmissão. É possível usar o campo de pesquisa para procurar um banco de dados pelo nome.
  - Table (Tabela): (opcional) escolha a tabela associada aos dados de origem na lista. Essa tabela já deve existir no AWS Glue Data Catalog. É possível usar o campo de pesquisa para procurar uma tabela pelo nome.
  - Detect schema (Detectar esquema): escolha essa opção para que o AWS Glue detecte o esquema dos dados de transmissão, em vez de armazenar as informações do esquema em uma tabela do Data Catalog. Essa opção é habilitada automaticamente quando a opção Stream details (Detalhes do fluxo) é escolhida.

Se você escolher Stream details (Detalhes do fluxo), especifique as informações adicionais a seguir.

- Connection name (Nome da conexão): escolha a conexão do AWS Glue que contém as informações de acesso e autenticação para o fluxo de dados do Kafka. Você deve usar uma conexão com origens de dados de streaming do Kafka. Se uma conexão não existe, é possível usar o console do AWS Glue para criar uma conexão para o fluxo de dados do Kafka.
- Topic name (Nome do tópico): insira o nome do tópico do qual a leitura será feita.
- Data format (Formato dos dados): escolha o formato a ser usado ao ler dados da sequência de eventos do Kafka.

- Starting position (Posição inicial): por padrão, o trabalho de ETL usa a opção Earliest (Mais antiga), o que significa que ele lê dados começando com o registro mais antigo disponível no fluxo. Em vez disso, você pode escolher Latest (Mais recente), o que indica que o trabalho de ETL deve começar a leitura logo após o registro mais recente no fluxo.
- Window size (Tamanho da janela): por padrão, o trabalho de ETL processa e grava dados em janelas de 100 segundos. Isso permite que os dados sejam processados de forma eficiente e que as agregações sejam realizadas em dados que chegam mais tarde do que o esperado. É possível modificar esse tamanho da janela para aumentar a pontualidade ou a precisão da agregação.

  Os trabalhos de transmissão do AWS Glue usam pontos de verificação em vez de marcadores de trabalho para rastrear os dados que foram lidos.
- Connection options (Opções de conexão): expanda essa seção para adicionar pares de chave-valor a fim de especificar opções de conexão adicionais. Para obter informações sobre quais opções você pode especificar aqui, consulte "connectionType": "kafka" no Guia do desenvolvedor do AWS Glue.

> **Note**
>
> As previsualizações de dados não são suportadas atualmente para fontes de dados de transmissão.

Referências

Melhores práticas

- Criar um pipeline de serviços de ETL para carregar dados incrementalmente do Amazon S3 ao Amazon Redshift usando AWS Glue

Programação ETL

- Tipos de conexão e opções para ETL no AWS Glue
- Valores de connectionType de JDBC
- Opções avançadas para mover dados de e para o Amazon Redshift

Adicionar uma conexão JDBC usando seus próprios drivers JDBC

Você pode usar seu próprio driver JDBC quando usar uma conexão JDBC. Quando o driver padrão utilizado pelo crawler do AWS Glue não consegue se conectar a um banco de dados, você pode usar seu próprio driver JDBC. Por exemplo, se quiser usar o SHA-256 com seu banco de dados Postgres e os drivers postgres mais antigos não forem compatíveis, você poderá usar seu próprio driver JDBC.

Fontes de dados compatíveis

Fontes de dados compatíveis	Fontes de dados não compatíveis
MySQL	Snowflake
Postgres	
Oracle	
Redshift	
SQL Server	
Aurora*	

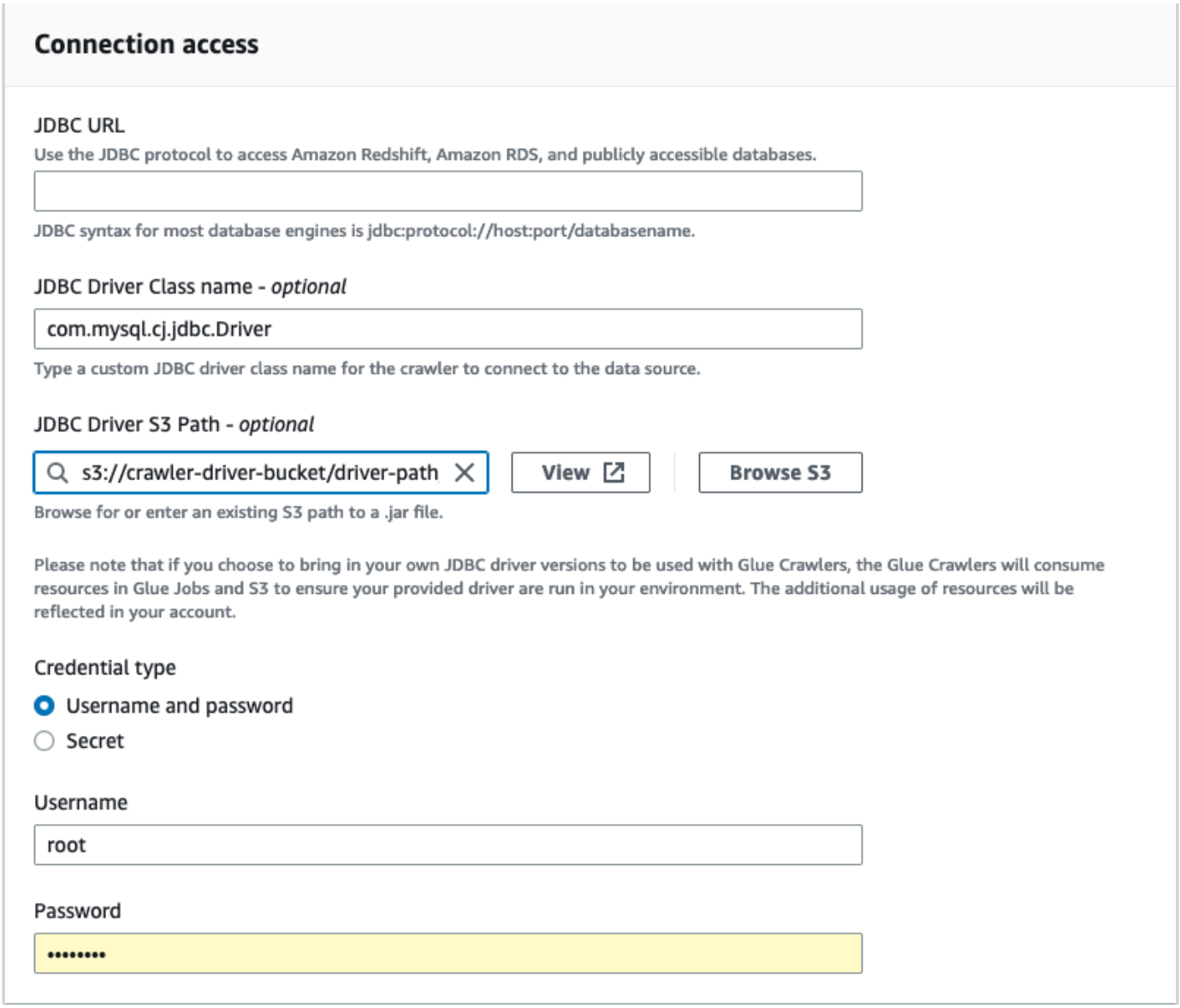
*Compatível se o driver JDBC nativo estiver sendo usado. Nem todos os recursos do driver podem ser aproveitados.

Adicionar um driver JDBC a uma conexão JDBC

> **Note**
>
> Se você optar por trazer suas próprias versões do driver JDBC, os crawlers do AWS Glue consumirão recursos em trabalhos do AWS Glue e buckets do Amazon S3 para garantir que o driver fornecido seja executado em seu ambiente. O uso adicional de recursos será refletido em sua conta. O custo dos crawlers e trabalhos do AWS Glue se enquadram na categoria do AWS Glue para cobrança. Além disso, fornecer seu próprio driver JDBC não significa que o crawler seja capaz de aproveitar todos os atributos do driver.

Para adicionar uma conexão JDBC a uma conexão JDBC:

1. Adicione o arquivo do driver JDBC a um local do Amazon S3. Você pode criar um bucket e/ou uma pasta ou usar os já existentes.
2. No console do AWS Glue, escolha Conexões no menu à esquerda em Catálogo de dados e crie uma nova conexão.
3. Preencha os campos para Propriedades da conexão e escolha JDBC como Tipo de conexão.
4. Em Acesso à conexão, insira a URL do JDBC e o Nome da classe do driver JDBC, opcional. O nome da classe do driver deve ser para uma fonte de dados compatível com os crawlers do AWS Glue.

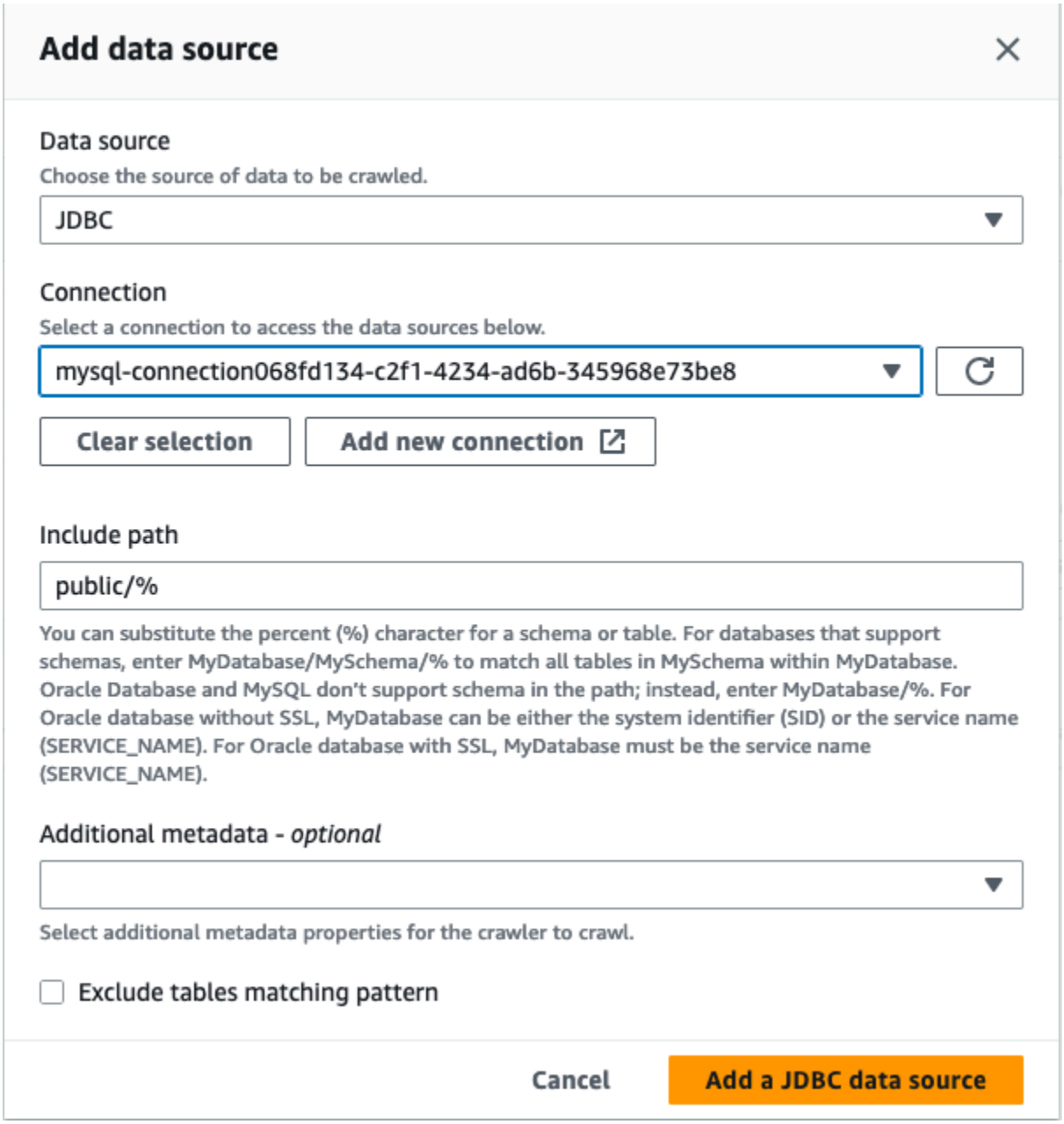


5. Escolha o caminho do Amazon S3 em que o driver JDBC está localizado em Caminho do Amazon S3 do driver JDBC, campo opcional.
6. Preencha os campos de Tipo de credencial se estiver inserindo um nome de usuário e senha ou segredo. Ao concluir, escolha Criar conexão.

> **Note**
>
> O teste de conexão não é compatível atualmente. Ao fazer crawling na fonte de dados com um driver JDBC que você forneceu, o crawler pula essa etapa.

7. Adicione a conexão recém-criada a um crawler. No console do AWS Glue, escolha Crawlers no menu esquerdo em Catálogo de dados e crie um novo crawler.

8. No assistente Adicionar crawler, na etapa 2, escolha Adicionar uma fonte de dados.



9. Escolha JDBC como fonte de dados e escolha a conexão que foi criada nas etapas anteriores. Concluído
10. Para usar seu próprio driver JDBC com um AWS Glue rastreador, adicione as seguintes permissões à função usada pelo rastreador:

    - Conceda permissões para as seguintes ações: `CreateJob`, `DeleteJob`, `GetJob`, `GetJobRun`, `StartJobRun`.
    - Conceda permissões para as ações do IAM: `iam:PassRole`

- Conceda permissões para as ações do Amazon S3: `s3:DeleteObjects`, `s3:GetObject`, `s3:ListBucket`, `s3:PutObject`.
- Conceda acesso de entidade principal de serviço a bucket/pasta na política do IAM.

Exemplo de política do IAM:

```
{
    "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. Se você estiver usando uma VPC, deverá permitir o acesso ao endpoint do AWS Glue criando o endpoint de interface e adicionando-o à sua tabela de rotas. Para obter mais informações, consulte Creating an interface VPC endpoint for AWS Glue
12. Se você estiver usando criptografia em seu catálogo de dados, crie o endpoint da AWS KMS interface e adicione-o à sua tabela de rotas. Para obter mais informações, consulte Creating a VPC endpoint for AWS KMS.

Testar uma conexão do AWS Glue

Como prática recomendada, antes de usar uma conexão do AWS Glue em um trabalho de ETL, use o console do AWS Glue para testar a conexão. O AWS Glue usa os parâmetros da conexão para

confirmar que pode acessar o datastore e relata qualquer erro encontrado. Para obter informações sobre as conexões do AWS Glue, consulte Conectar a dados.

Como testar uma conexão do AWS Glue

1. Faça login no AWS Management Console e abra o AWS Glue console em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Catálogo de dados, selecione Conexões. Você também pode escolher Conexões de dados acima de Catálogo de dados no painel de navegação.
3. Em Conexões, marque a caixa de seleção ao lado da conexão desejada e selecione Ações. No menu suspenso , escolha Testar conexão.
4. Na caixa de diálogo Testar conexão, selecione uma função ou escolha Criar função do IAM para acessar o console AWS Identity and Access Management (IAM) e criar uma nova função. A função deve ter permissões no datastore.
5. Selecione a opção Confirmar.

   O teste é iniciado e pode levar vários minutos para ser concluído. Se o teste falhar, escolha Solucionar problemas para ver as etapas para resolver o problema.
6. Escolha Registros para ver os registros CloudWatch. É necessário ter as permissões do IAM exigidas para visualizar os logs. Para obter mais informações, consulte Políticas AWS gerenciadas (predefinidas) para CloudWatch registros no Guia do usuário do Amazon CloudWatch Logs.

Configurar chamadas da AWS para passar por sua VPC

O parâmetro especial de trabalho `disable-proxy-v2` permite rotear suas chamadas para serviços como Amazon S3, CloudWatch e AWS Glue por meio da sua VPC. Por padrão, o AWS Glue usa um proxy local para enviar tráfego pela VPC do AWS Glue para baixar scripts e bibliotecas do Amazon S3, enviar solicitações ao CloudWatch para publicação de logs e métricas, e para enviar ao AWS Glue solicitações de acesso a catálogos de dados. Esse proxy permite que o trabalho funcione normalmente, mesmo se a VPC não configurar uma rota adequada para outros serviços da AWS, como o Amazon S3, o CloudWatch e o AWS Glue. O AWS Glue agora oferece um parâmetro para você desativar esse comportamento. Para obter mais informações, consulte Parâmetros de trabalho usados pelo AWS Glue. O AWS Glue continuará a usar o proxy local para publicar logs do CloudWatch de seus trabalhos do AWS Glue.

> **Note**
>
> - Esse recurso é compatível com trabalhos do AWS Glue com o AWS Glue versão 2.0 e superior. Ao usar esse recurso, você precisa garantir que sua VPC tenha configurado uma rota para o Amazon S3 por meio de um endpoint de NAT ou VPC de serviço.
> - O parâmetro de trabalho obsoleto `disable-proxy` só encaminha as chamadas para o Amazon S3 para baixar scripts e bibliotecas pela VPC. É recomendável usar o novo parâmetro `disable-proxy-v2` em seu lugar.

Exemplo de uso

Criar um trabalho do AWS Glue com `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"}'
```

Conectar-se a um armazenamento de dados JDBC em uma VPC

Geralmente, você cria esses recursos dentro da Amazon Virtual Private Cloud (Amazon VPC), para que eles não possam ser acessados pela Internet pública. Por padrão, o AWS Glue não pode acessar recursos dentro de uma VPC. Para permitir que o AWS Glue acesse recursos dentro da sua VPC, é preciso fornecer informações adicionais de configuração específicas da VPC, incluindo IDs de sub-rede da VPC e IDs de grupos de segurança. O AWS Glue usa essas informações para configurar interfaces de rede elásticas que permitem que sua função se conecte com segurança a outros recursos na sua VPC privada.

Ao usar um endpoint da VPC, adicione-o à sua tabela de rotas. Para obter mais informações, consulte Creating an interface VPC endpoint for AWS Glue e Pré-requisitos.

Ao usar a criptografia no catálogo de dados, crie o endpoint da interface KMS e adicione-o à sua tabela de rotas. Para obter mais informações, consulte Creating a VPC endpoint for AWS KMS.

Acessar dados da VPC usando interfaces de rede elástica

Quando o AWS Glue se conecta a um armazenamento de dados JDBC em uma VPC, o AWS Glue cria uma interface de rede elástica (com o prefixo `Glue_`) na sua conta para acessar seus dados na VPC. Não será possível excluir essa interface de rede enquanto ela estiver vinculada ao AWS Glue. Como parte da criação da interface de rede elástica, o AWS Glue associa um ou mais grupos de segurança a ela. Para permitir que o AWS Glue crie a interface de rede, os grupos de segurança associados ao recurso precisam conceder acesso de entrada com uma regra de origem. Esta regra contém um security group associado ao recurso. Isso concede acesso para a interface de rede elástica ao seu armazenamento de dados que contém o mesmo security group.

Para permitir que o AWS Glue se comunique com seus respectivos componentes, especifique um grupo de segurança com uma regra de entrada de autorreferência para todas as portas TCP. Ao criar uma regra de autorreferência, você pode restringir a origem ao mesmo security group na VPC e não abri-la para todas as redes. O security group padrão para sua VPC pode já conter uma regra de entrada de autorreferenciada para `ALL Traffic`.

Você pode criar regras no console do Amazon VPC. Para atualizar as configurações de regra por meio do AWS Management Console, navegue até o console da VPC (https://console.aws.amazon.com/vpc/) e selecione o grupo de segurança apropriado. Especifique a regra de entrada para `ALL TCP` para que a fonte tenha o mesmo nome do security group. Para obter mais informações sobre regras de grupos de segurança, consulte Grupos de segurança para sua VPC.

Cada interface de rede elástica recebe um endereço IP privado do intervalo de endereços IP nas sub-redes que você especifica. A interface de rede não recebe qualquer endereço IP público atribuído. O AWS Glue requer acesso à Internet (por exemplo, para acessar os produtos da AWS que não têm endpoints de VPC). Você pode configurar uma instância de conversão de endereço de rede (NAT) na sua VPC ou usar o gateway NAT da Amazon VPC. Para obter mais informações, consulte Gateways NAT no Manual do usuário da Amazon VPC. Não é possível usar um gateway da Internet diretamente vinculado à sua VPC como uma tabela de rotas da sua sub-rede porque isso requer que a interface de rede contenha endereços IP públicos.

Os atributos de rede `enableDnsHostnames` e `enableDnsSupport` da VPC precisam ser definidos como true. Para obter mais informações, consulte Como usar o DNS com sua VPC.

> **Important**
>
> Não coloque seu armazenamento de dados em uma sub-rede pública nem em uma sub-rede privada que não tenha acesso à Internet. Em vez disso, anexe-o somente a sub-redes privadas com acesso à Internet por meio de uma instância NAT ou de um gateway NAT da Amazon VPC.

Propriedades da interface de rede elástica (ENI)

Para criar a interface de rede elástica, você precisa fornecer as seguintes propriedades:

VPC

O nome da VPC que contém seu armazenamento de dados.

Sub-rede

A sub-rede na VPC que contém seu armazenamento de dados.

Grupos de segurança

Os grupos de segurança associados ao seu armazenamento de dados. O AWS Glue associa esses grupos de segurança à interface de rede elástica anexada à sua sub-rede VPC. Para permitir que os componentes do AWS Glue se comuniquem e também para impedir o acesso de outras redes, pelo menos um grupo de segurança escolhido deve especificar uma regra de entrada de autorreferência para todas as portas TCP.

Para obter informações sobre como gerenciar uma VPC com o Amazon RedShift, consulte Gerenciar clusters em uma Amazon Virtual Private Cloud (VPC).

Para obter informações sobre como gerenciar uma VPC com o Amazon Relational Database Service (Amazon RDS), consulte Trabalhar com uma instância de banco de dados do Amazon RDS em uma VPC.

Usar uma conexão do MongoDB ou do MongoDB Atlas

Após criar uma conexão para o MongoDB ou para o MongoDB Atlas, você poderá usá-la em seu trabalho de ETL. Você cria uma tabela no AWS Glue Data Catalog e especifica a conexão do MongoDB ou do MangoDB Atlas para o atributo `connection` da tabela.

O AWS Glue armazena o `url` de sua conexão e credenciais na conexão do MongoDB. Os formatos de URI de conexão são os seguintes:

- Para o MongoDB: mongodb://host:port/database. O host pode ser um nome de host, um endereço IP ou um soquete de domínio do UNIX. Se a string de conexão não especificar uma porta, a porta padrão do MongoDB, 27017, será usada.
- Para o MongoDB Atlas: mongodb+srv://server.example.com/database. O host pode ser um nome de host que corresponde a um registro SRV do DNS. O formato SRV não requer porta e usará a porta 27017, padrão do MongoDB.

Além disso, é possível especificar opções em seu script de trabalho. Para ter mais informações, consulte the section called “Conexão do MongoDB”.

Crawling em armazenamento de dados do Amazon S3 usando um endpoint da VPC

Para fins de segurança, auditoria ou controle, talvez você queira que seu armazenamento de dados do Amazon S3 ou suas tabelas de catálogo de dados baseadas no Amazon S3 sejam acessados somente por meio de um ambiente do Amazon Virtual Private Cloud (Amazon VPC). Este tópico descreve como criar e testar uma conexão com o armazenamento de dados do Amazon S3 ou com tabelas de catálogo de dados baseadas no Amazon S3 em um endpoint da VPC usando o tipo de conexão `Network`.

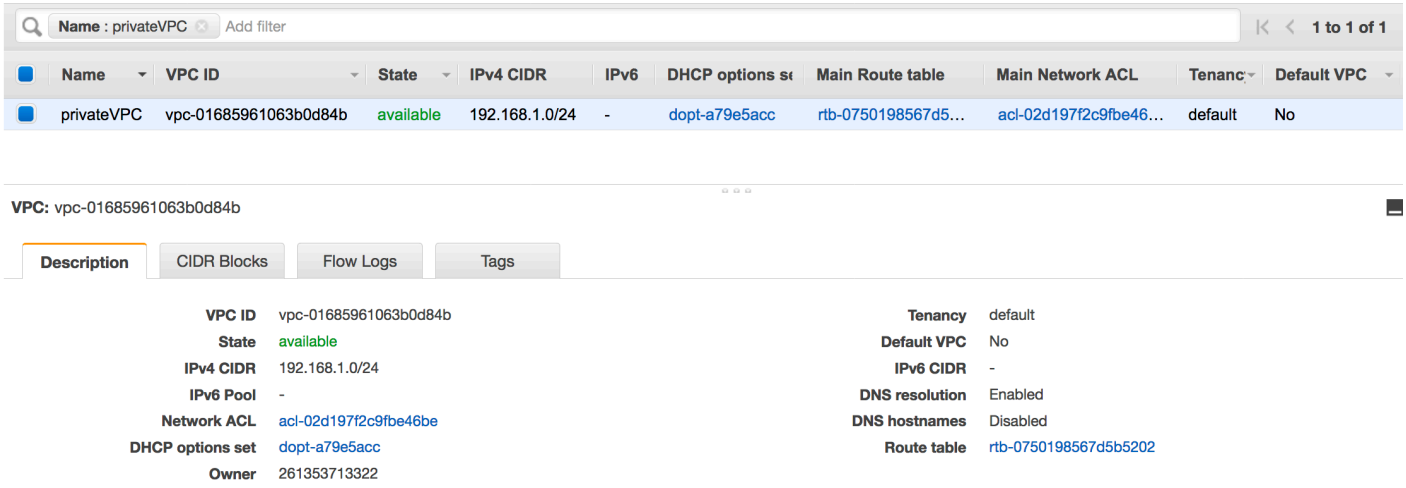
Realize as seguintes tarefas para executar um crawler no armazenamento de dados:

- the section called “Pré-requisitos”
- the section called “Criar a conexão com o Amazon S3”
- the section called “Testar a conexão com o Amazon S3”
- the section called “Criar um crawler para um armazenamento de dados do Amazon S3”
- the section called “Executar um crawler”

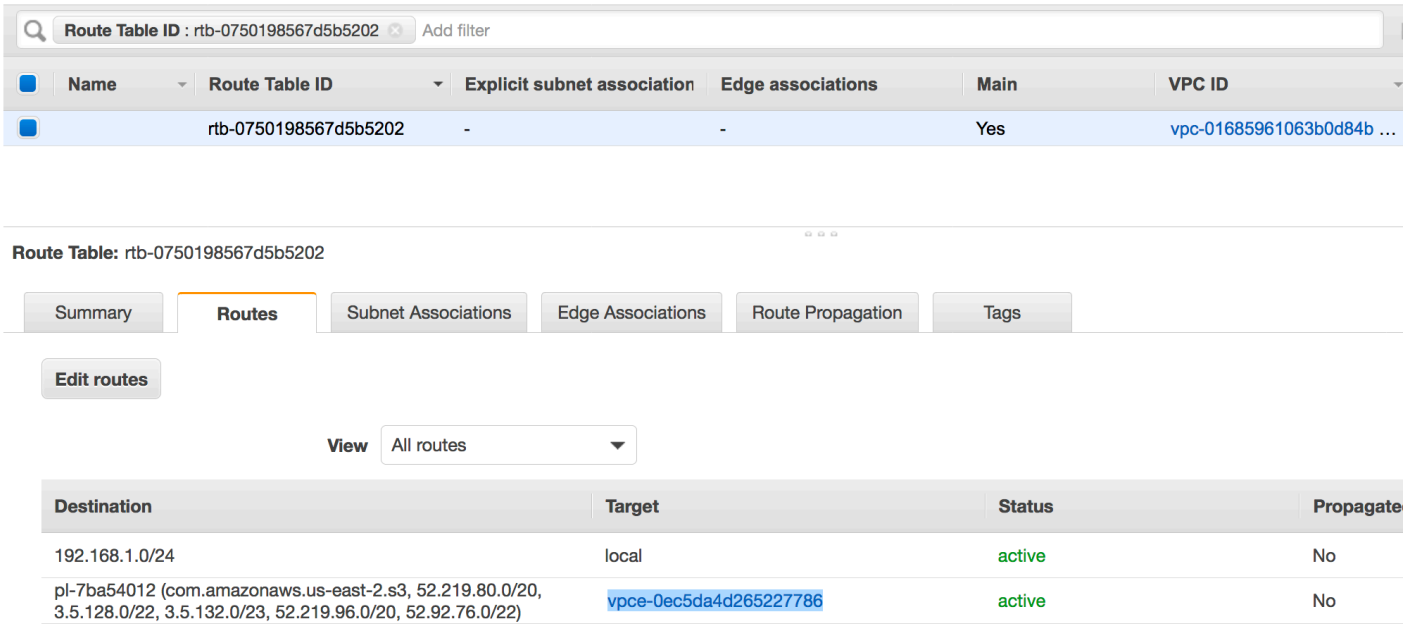
Pré-requisitos

Verifique se você atendeu a esses pré-requisitos para configurar o armazenamento de dados do Amazon S3 ou as tabelas de catálogo de dados baseadas no Amazon S3 para serem acessados por meio de um ambiente da Amazon Virtual Private Cloud (Amazon VPC).

- Uma VPC configurada. Por exemplo: vpc-01685961063b0d84b. Para obter mais informações, consulte Conceitos básicos da Amazon VPC no Manual do usuário da Amazon VPC.
- Um endpoint do Amazon S3 anexado à VPC. Por exemplo: vpc-01685961063b0d84b. Para obter mais informações, consulte Endpoints para o Amazon S3 no Manual do usuário da Amazon VPC.



- Uma entrada de rota apontando para o endpoint da VPC. Por exemplo: vpce-0ec5da4d265227786 na tabela de rotas usada pelo endpoint da VPC (vpce-0ec5da4d265227786).



- Uma ACL da rede anexada à VPC permite o tráfego.
- Um grupo de segurança anexado à VPC permite o tráfego.

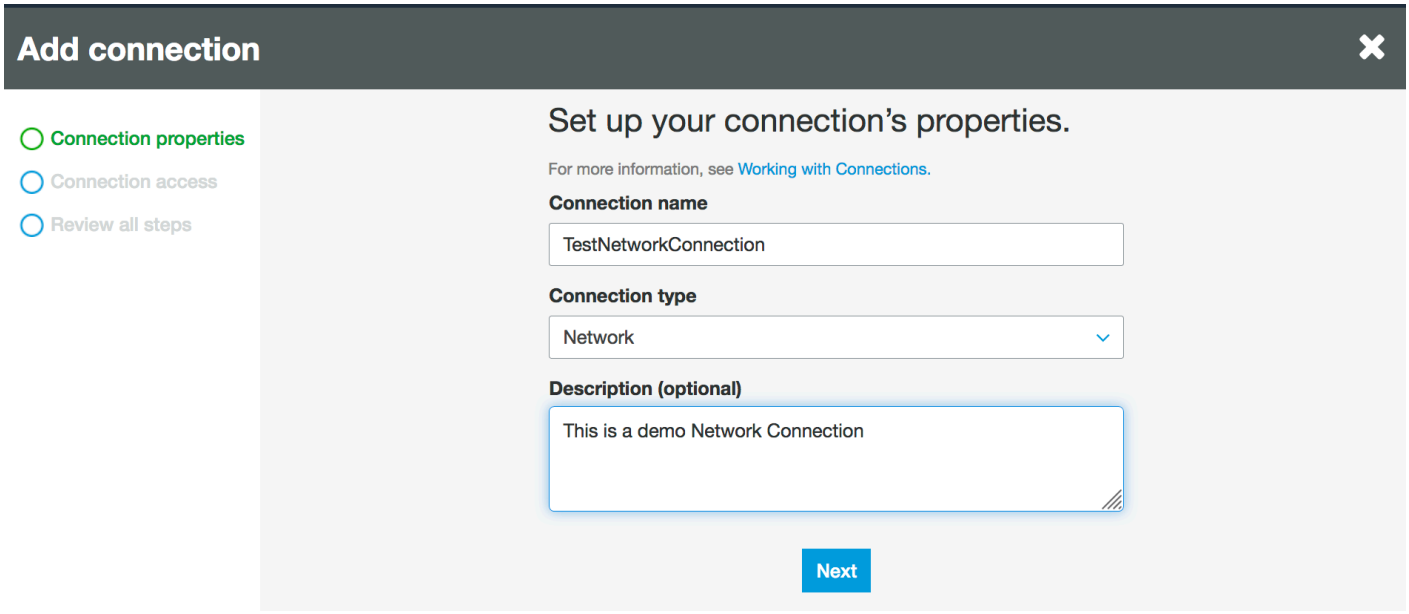
Criar a conexão com o Amazon S3

Geralmente, você cria esses recursos dentro da Amazon Virtual Private Cloud (Amazon VPC), para que eles não possam ser acessados pela Internet pública. Por padrão, o AWS Glue não pode acessar recursos dentro de uma VPC. Para permitir que o AWS Glue acesse seus recursos dentro da VPC, é preciso fornecer informações adicionais de configuração específicas da VPC que incluem IDs de sub-redes da VPC e IDs de security groups. Para criar uma conexão `Network`, você precisa especificar as seguintes informações:

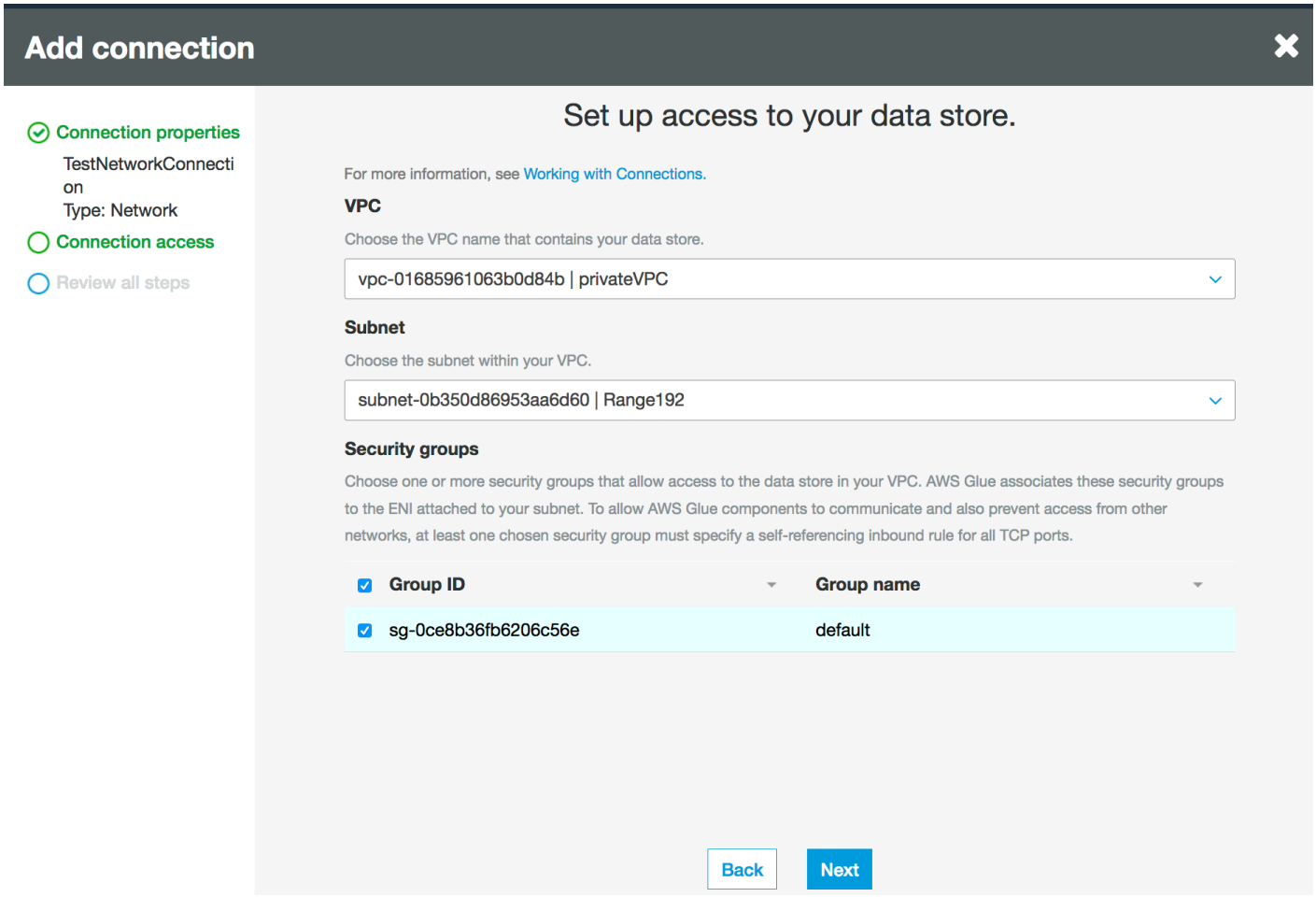
- Uma ID da VPC
- Uma sub-rede dentro da VPC
- Um grupo de segurança

Para configurar uma conexão `Network`:

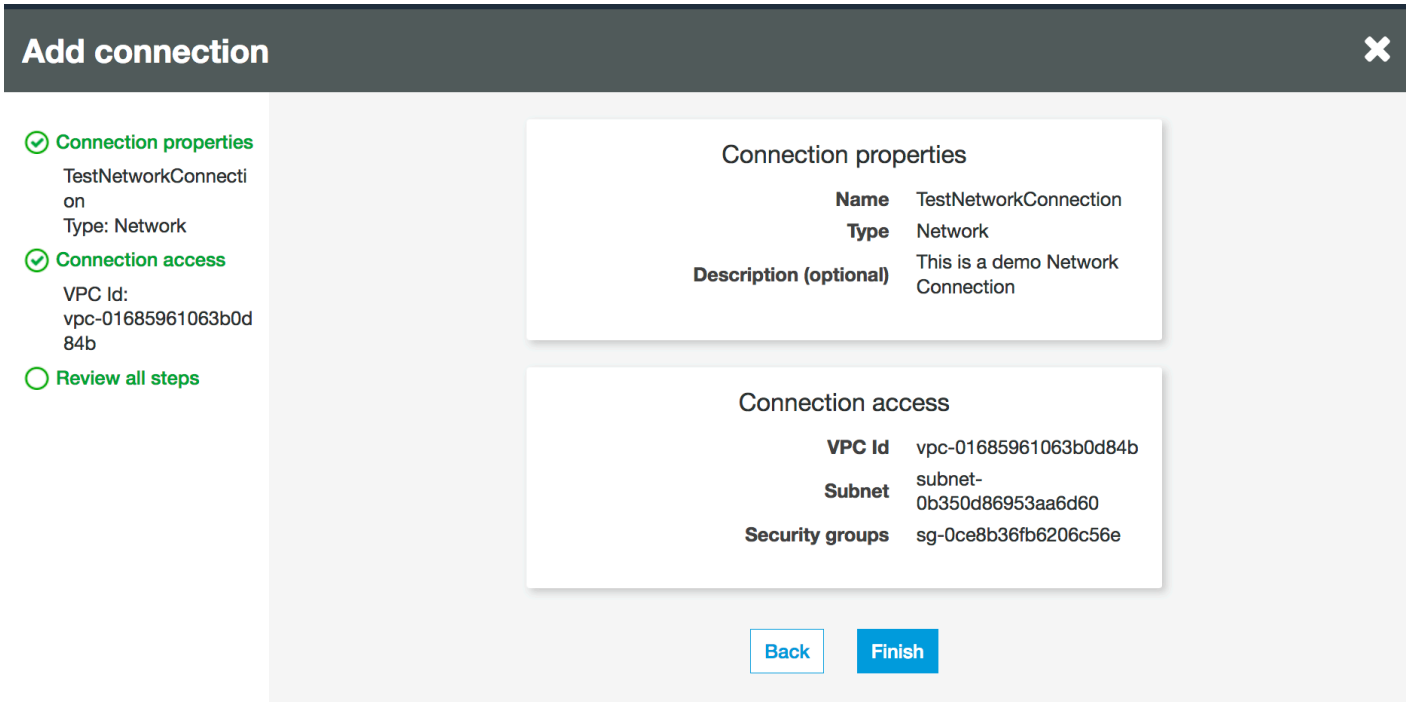
1. Escolha Add connection (Adicionar conexão) no painel de navegação do console do AWS Glue.
2. Insira o nome da conexão e escolha Network (Rede) como o tipo de conexão. Escolha Próximo.



3. Configure as informações de VPC, sub-rede e grupos de segurança.
   - VPC: escolha o nome da VPC que contém seu armazenamento de dados.
   - Subnet (Sub-rede): escolha uma sub-rede em sua VPC.
   - Security groups (Grupos de segurança): escolha um ou mais grupos de segurança que permitam o acesso ao armazenamento de dados em sua VPC.



4. Escolha Próximo.
5. Verifique as informações de conexão e escolha Finish (Encerrar).



Testar a conexão com o Amazon S3

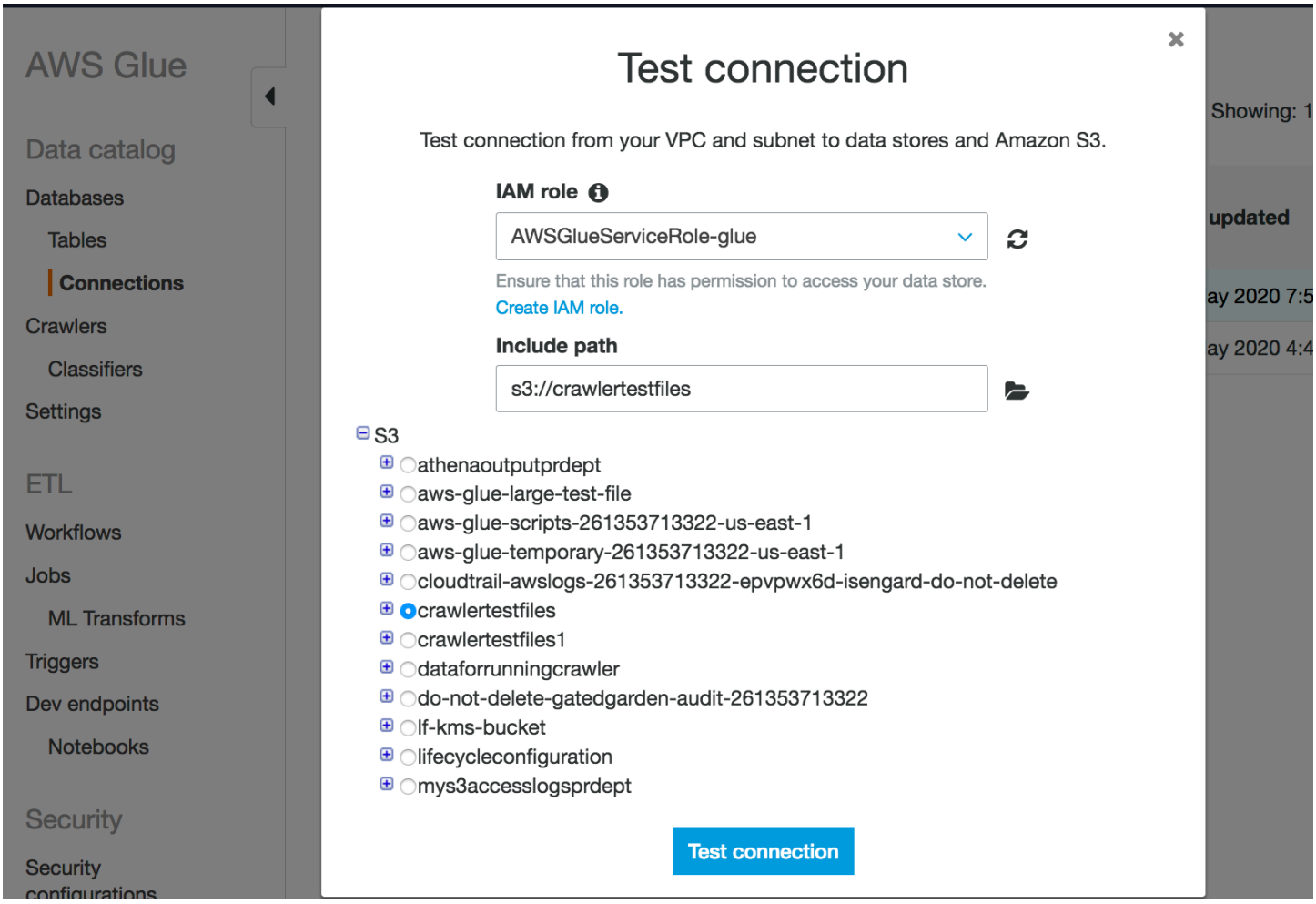
Depois de criar sua conexão `Network`, você pode testar a conectividade com seu armazenamento de dados do Amazon S3 em um endpoint da VPC.

Os seguintes erros podem ocorrer ao testar uma conexão:

- INTERNET CONNECTION ERROR (Erro de conexão com a Internet): indica um problema de conexão
- INVALID BUCKET ERROR (Erro de bucket inválido): indica um problema com o bucket do Amazon S3
- Se CONNECTION ERROR (Erro de conexão com o S3): indica uma falha na conexão com o Amazon S3
- INVALID CONNECTION TYPE (Tipo de conexão inválida): indica que o tipo de conexão não tem o valor esperado, `NETWORK`
- INVALID CONNECTION TEST TYPE (Tipo teste de conexão inválido): indica um problema com o tipo de teste de conexão de rede
- INVALID TARGET (Destino inválido): indica que o bucket do Amazon S3 não foi especificado corretamente

Para testar uma conexão `Network`:

1. Selecione a conexão Network (Rede) no console do AWS Glue.
2. Selecione Test connection (Testar conexão).
3. Escolha a função do IAM criada na etapa anterior e especifique um bucket do Amazon S3.
4. Escolha Test connection (Testar conexão) para iniciar o teste. Pode levar algum tempo para que o resultado seja exibido.



Se você receber um erro, faça o seguinte:

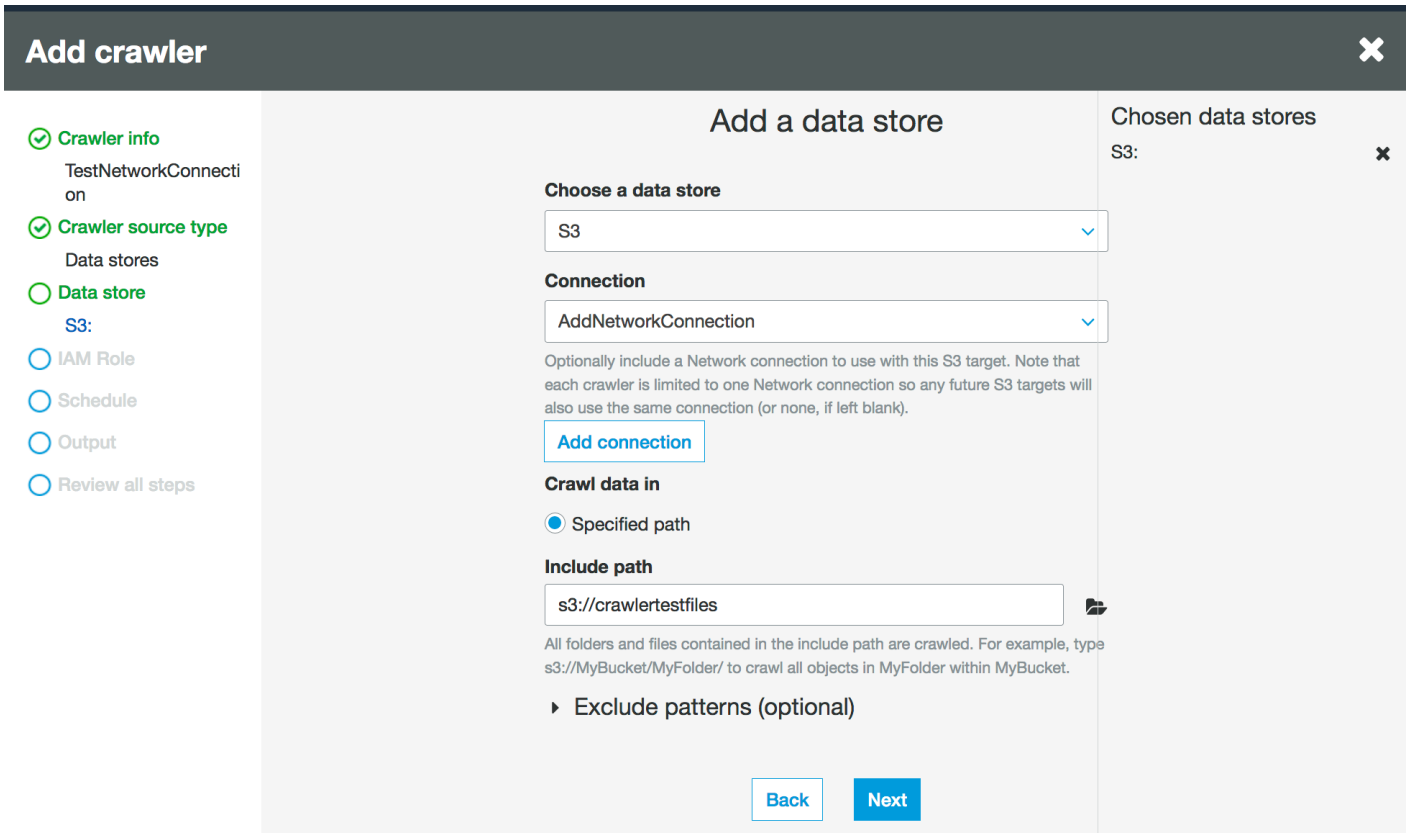
- Os privilégios corretos foram fornecidos para a função selecionada.
- O bucket do Amazon S3 correto foi fornecido.
- Os grupos de segurança e a ACL da rede permitem o tráfego de entrada e saída necessário.
- A VPC especificada está conectada a um endpoint da VPC do Amazon S3.

Após ter testado com êxito a conexão, você pode criar um crawler.

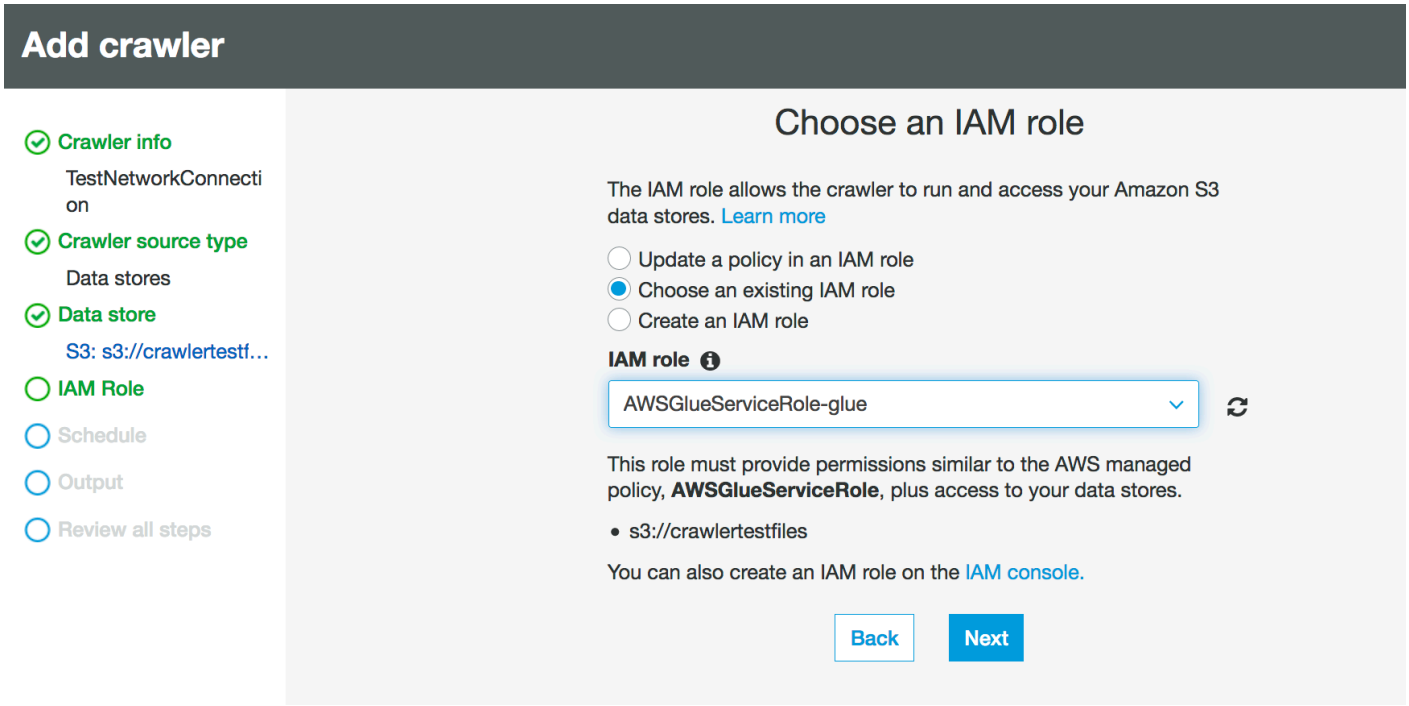
Criar um crawler para um armazenamento de dados do Amazon S3

Agora é possível criar um crawler que especifica a conexão `Network` que você criou. Para obter mais detalhes sobre como criar um crawler, consulte Configurar um crawler.

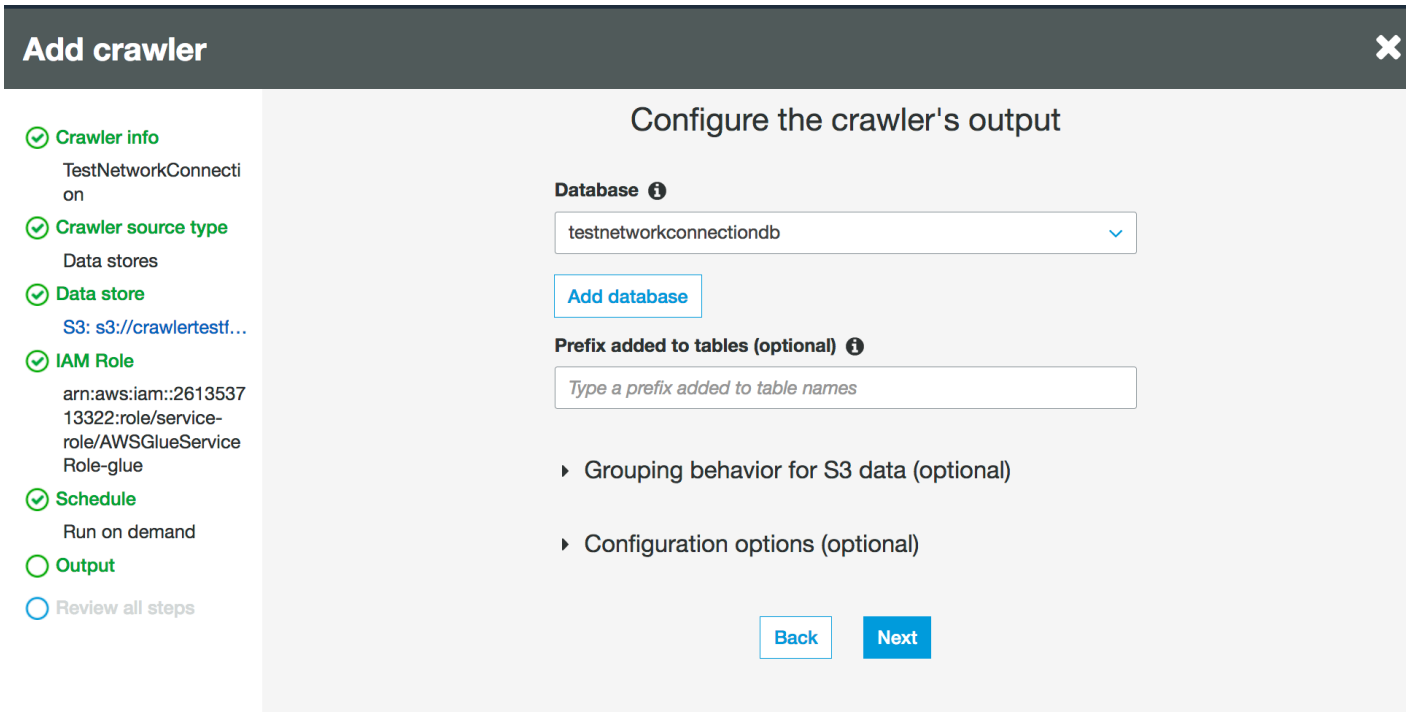
1. Comece escolhendo Crawlers no painel de navegação no console do AWS Glue.
2. Escolha Adicionar crawler.
3. Especifique o nome do crawler e escolha Next (Próximo).
4. Quando a origem dos dados for solicitada, escolha S3 e especifique o prefixo do bucket do Amazon S3 e a conexão criada anteriormente.



5. Se precisar, adicione outro armazenamento de dados na mesma conexão de rede.
6. Escolha a função do IAM. A função do IAM deve permitir o acesso ao serviço do AWS Glue e o bucket do Amazon S3. Para ter mais informações, consulte the section called “Configurar um crawler”.



7. Defina a programação do crawler.
8. Escolha um banco de dados existente no Data Catalog ou crie uma nova entrada de banco de dados.

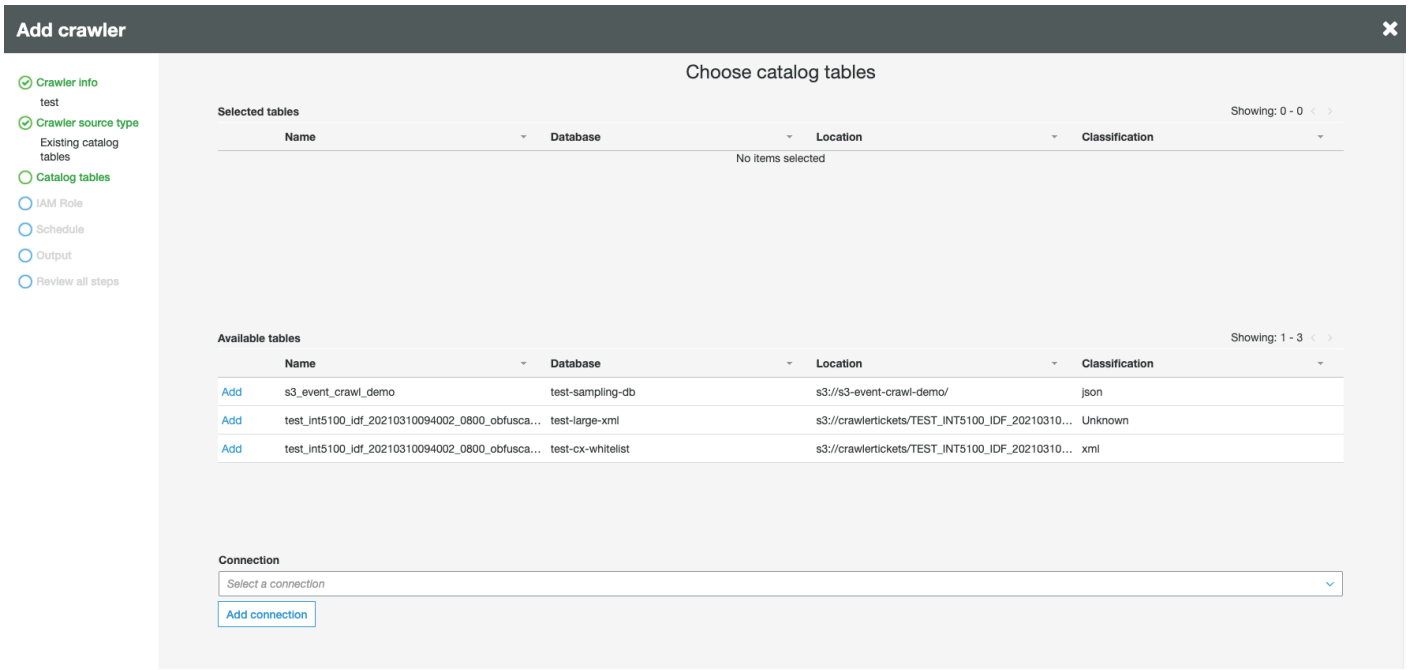


9. Conclua a configuração restante.

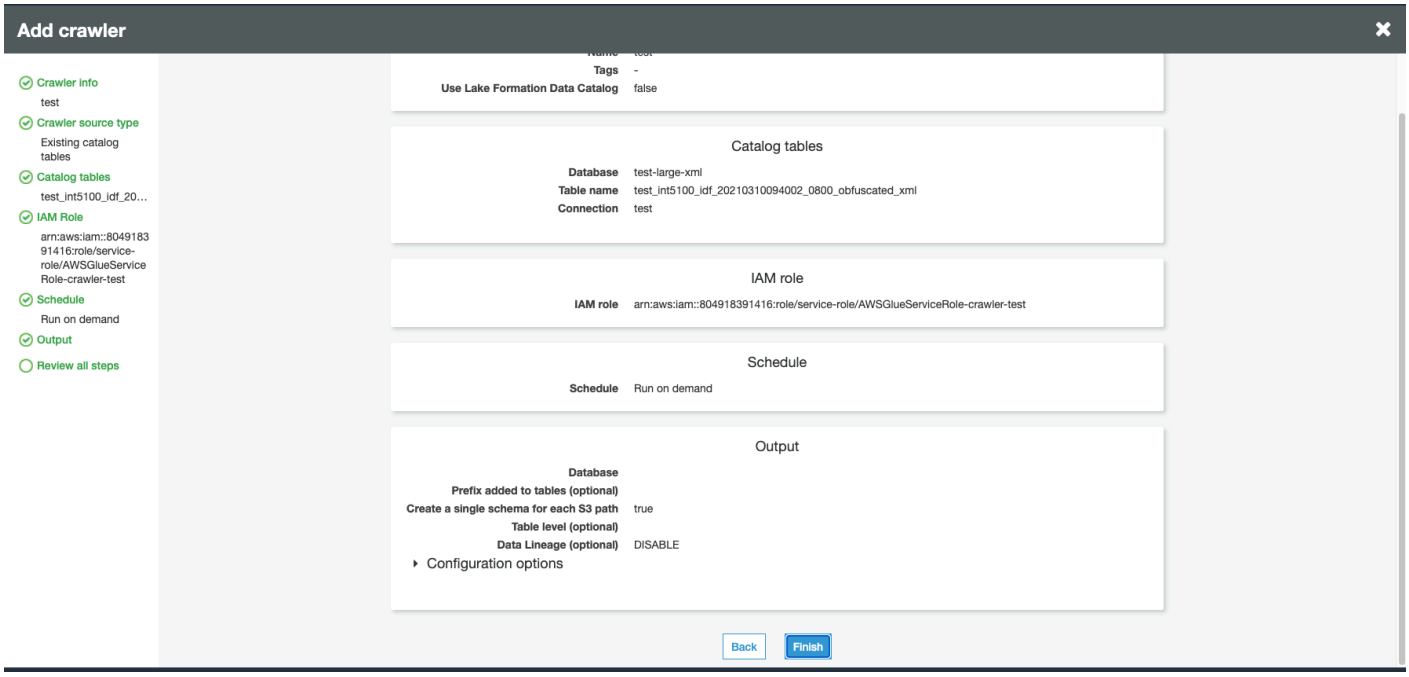
Criação de um crawler para tabelas de catálogo de dados baseadas no Amazon S3

Agora é possível criar um crawler que especifica a conexão de `Network` que você criou e um tipo de fonte de catálogo. Para obter mais detalhes sobre como criar um crawler, consulte Configurar um crawler.

1. Comece escolhendo Crawlers no painel de navegação no console do AWS Glue.
2. Escolha Adicionar crawler.
3. Especifique o nome do crawler e escolha Next (Próximo).
4. Quando o tipo de fonte do crawler for solicitado, escolha Existing catalog tables (Tabelas de catálogo existentes) e especifique as tabelas de catálogo existentes para crawling na lista de tabelas disponíveis.

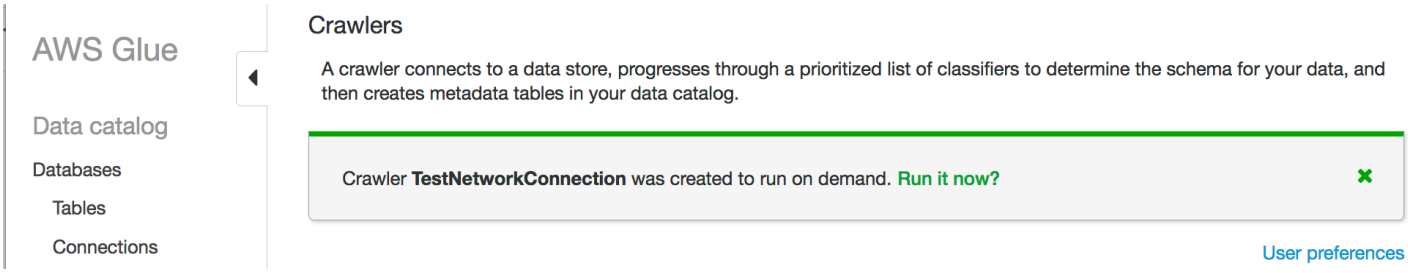


5. Escolha a função do IAM. A função do IAM deve permitir o acesso ao serviço do AWS Glue e o bucket do Amazon S3. Para ter mais informações, consulte the section called “Configurar um crawler”.
6. Defina a programação do crawler.
7. Escolha um banco de dados existente no Data Catalog ou crie uma nova entrada de banco de dados.
8. Conclua a configuração restante e revise suas etapas.



Executar um crawler

Execute seu crawler.



Solução de problemas

Para solucionar problemas relacionados aos buckets do Amazon S3 usando um gateway da VPC, consulte Por que não consigo me conectar a um bucket do S3 usando um endpoint da VPC de gateway?

Solução de problemas de conexão no AWS Glue

Quando um crawler ou um trabalho do AWS Glue usa propriedades de conexão para acessar um armazenamento de dados, você pode encontrar erros ao tentar se conectar. O AWS Glue usa endereços IP privados na sub-rede ao criar interfaces de rede elástica na nuvem privada virtual

(VPC) e sub-rede especificadas. Grupos de segurança especificados na conexão do são aplicados a cada uma das interfaces de rede elásticas. Verifique se os security groups concedem acesso de saída e permitem conectividade com o cluster de banco de dados.

Além disso, o Apache Spark exige conectividade bidirecional entre os nós do driver e do executor. Um dos security groups precisa permitir regras de entrada em todas as portas TCP. Você pode impedir que o security group seja aberto ao público, restringindo sua fonte para ele próprio com um security group de autorreferência.

Veja a seguir algumas ações típicas que você pode realizar para solucionar problemas de conexão:

- Verifique o endereço da porta da sua conexão.
- Verifique a string de nome de usuário e senha na sua conexão ou segredo.
- Para um armazenamento de dados JDBC, verifique se ele permite conexões de entrada.
- Verifique se seu armazenamento de dados pode ser acessado dentro da VPC.
- Se você armazenar suas credenciais de conexão usando o AWS Secrets Manager, verifique se o perfil do IAM para o AWS Glue tem permissão para acessar seu segredo. Para obter mais informações, consulte Exemplo: permissão para recuperar valores de segredos no Guia do usuário do AWS Secrets Manager. Dependendo da sua configuração de rede, talvez também seja necessário criar um endpoint da VPC para estabelecer uma conexão privada entre sua VPC e o Secrets Manager. Para obter mais informações, consulte Usar um endpoint da VPC no AWS Secrets Manager.

Tutorial: Usar o AWS Glue Connector for Elasticsearch

O Elasticsearch é um conhecido mecanismo de pesquisa e análise de código aberto para casos de uso, como análise de logs, monitoramento de aplicações em tempo real e análise de transmissões de cliques. Você pode usar o OpenSearch como um armazenamento de dados para seus trabalhos de extração, transformação e carregamento (ETL) configurando o AWS Glue Connector for Elasticsearch no AWS Glue Studio. Esse conector está disponível gratuitamente no AWS Marketplace.

> **Note**
>
> O AWS Marketplace Elasticsearch Spark Connector foi descontinuado. Use o AWS Glue Connector for Elasticsearch em seu lugar.

Neste tutorial, mostraremos como se conectar aos nós do Amazon OpenSearch Service com um número mínimo de etapas.

Tópicos

Pré-requisitos

Para usar este tutorial, você deve ter o seguinte:

- Acessar o AWS Glue Studio
- Acesso a um cluster do OpenSearch na Nuvem AWS
- (Opcional) Acesso ao AWS Secrets Manager.

Etapa 1: (opcional) Criar um segredo da AWS para as informações do cluster do OpenSearch

Para armazenar e usar com segurança sua credencial de conexão, salve-a no AWS Secrets Manager. O segredo que você criar será usado pela conexão, mais tarde no tutorial. Os pares chave-valor de credenciais serão alimentados no AWS Glue Connector for Elasticsearch como opções de conexão normais.

Para obter mais informações sobre criação de segredos, consulte Criar e gerenciar segredos com o AWS Secrets Manager no Manual do usuário do AWS Secrets Manager.

Para criar um segredo da AWS

1. Faça login no AWS Secrets Managerconsole.
2. Na página de introdução do serviço ou na página da lista de Secrets (Segredos), escolha Store a new secret (Armazenar um novo segredo).

3. Na página Store a new secret (Armazenar um novo segredo), selecione Other type of secret (Outro tipo de segredo). Essa opção significa que você precisa fornecer a estrutura e os detalhes do seu segredo.
4. Adicione um par de Key (Chave) e Value (Valor) para o nome de usuário do cluster do OpenSearch. Por exemplo:

   `es.net.http.auth.user`: *`nome de usuário`*
5. Escolha + Add row (Adicionar linha) e insira outro par de chave-valor para a senha. Por exemplo:

   `es.net.http.auth.pass`: *`senha`*
6. Escolha Próximo.
7. Insira um nome para o segredo. Por exemplo: meu-segredo-es. Se preferir, você também pode inserir uma descrição.

   Registre o nome secreto, que é usado posteriormente neste tutorial, e escolha Next (Próximo).
8. Escolha Next (Próximo) novamente e, depois, escolha Store (Armazenar) para criar o segredo.

Próxima etapa

Etapa 2: Assinar o conector

Etapa 2: Assinar o conector

O AWS Glue Connector for Elasticsearch está disponível gratuitamente no AWS Marketplace.

Para assinar o AWS Glue Connector for Elasticsearch no AWS Marketplace

1. Se você ainda não configurou a conta da AWS para usar o License Manager, faça o seguinte:
   a. Abra o console AWS License Manager em https://console.aws.amazon.com/license-manager.
   b. Escolha Create customer managed license (Criar licença gerenciada pelo cliente).
   c. Na janela Permissões do IAM (configuração única), escolha Concedo as permissões necessárias ao AWS License Manager e depois escolha Conceder permissões.

      Se você não vir essa janela, você já configurou as permissões necessárias.
2. Abra o console do AWS Glue Studio em https://console.aws.amazon.com/gluestudio/.

3. No console do AWS Glue Studio, expanda o ícone de menu (☰) e escolha Connectors (Conectores) no painel de navegação.
4. Na página Conectores, escolha Ir para o AWS Marketplace.
5. No AWS Marketplace, na seção Pesquisar produtos da AWS Glue Studio, insira Conector do AWS Glue para Elasticsearch no campo de pesquisa e pressione Enter.
6. Escolha o nome do conector, AWS Glue Connector for Elasticsearch.
7. Na página de produto do conector, use as guias para visualizar informações sobre o conector. Quando estiver tudo pronto para continuar, escolha Continue to Subscribe (Avançar para assinar).
8. Revise e aceite os termos de uso do software. Clique em Accept Terms (Aceitar os termos).
9. Quando o processo de assinatura for concluído, você verá uma notificação: "Thank you for subscribing to this product! You can now configure your software" (Obrigado por assinar este produto! Agora você pode configurar o software). Acima do banner, encontra-se o botão Continue to Configuration(Continuar em Configuração). Escolha Continue to Configuration (Continuar para configuração).
10. Escolha a opção Fulfillment (Atendimento) na página Configure this software (Configurar este software). Você também pode escolher o AWS Glue 1.0/2.0 ou o AWS Glue 3.0. Em seguida, escolha Continue to Launch (Continuar em Iniciar).

Próxima etapa

Etapa 3: Ativar o conector no AWS Glue Studio e criar uma conexão

Etapa 3: Ativar o conector no AWS Glue Studio e criar uma conexão

Depois de escolher Continue to Launch (Avançar para iniciar), você verá a página Launch this software (Iniciar este software) no AWS Marketplace. Depois de usar o link para ativar o conector no AWS Glue Studio, você cria uma conexão.

Para implantar o conector e criar uma conexão no AWS Glue Studio

1. Na página Launch this software (Iniciar este software) no console do AWS Marketplace, escolha Usage Instructions (Instruções de uso) e, em seguida, selecione o link na janela exibida.

   Seu navegador é redirecionado para a página Create marketplace connection (Criar conexão com o marketplace) do console do AWS Glue Studio.

2. Insira um nome para a conexão. Por exemplo: minha-conexão-es.
3. Na seção Connection access (Acesso à conexão), em Connection credential type (Tipo de credencial de conexão), escolha User name and password (Nome de usuário e senha).
4. Em segredo da AWS, insira o nome do seu segredo. Por exemplo: meu-segredo-es.
5. Na seção Network options (Opções de rede), insira as informações da VPC para conectar ao cluster do OpenSearch.
6. Esoclha Create connection and activate connector (Criar conexão e ativar o conector).

Próxima etapa

Etapa 4: Configurar uma função do IAM para o trabalho de ETL

Etapa 4: Configurar uma função do IAM para o trabalho de ETL

Ao criar o trabalho de ETL do AWS Glue, você especifica uma função do AWS Identity and Access Management (IAM) para o trabalho usar. A função deve conceder acesso a todos os recursos usados pelo trabalho, incluindo o Amazon S3 (para quaisquer origens, destinos, scripts, arquivos de driver e diretórios temporários) e também objetos do AWS Glue Data Catalog.

A função do IAM assumida para o trabalho de ETL do AWS Glue também deve ter acesso ao segredo que foi criado na seção anterior. Por padrão, a função gerenciada pela AWS `AWSGlueServiceRole` não tem acesso ao segredo. Para configurar o controle de acesso a seus segredos, consulte Autenticação e controle de acesso para o AWS Secrets Manager e Limitar acesso a segredos específicos.

Para configurar uma função do IAM para o trabalho de ETL

1. Configure as permissões descritas em the section called “Revisar as permissões do IAM necessárias para trabalhos de ETL”.
2. Configure as permissões adicionais necessárias ao usar conectores com o AWS Glue Studio, conforme descrito em the section called “Permissões necessárias para usar conectores”.

Próxima etapa

Etapa 5: Criar um trabalho que usa a conexão do OpenSearch

Etapa 5: Criar um trabalho que usa a conexão do OpenSearch

Depois de criar uma função para o seu trabalho de ETL, você pode criar um trabalho no AWS Glue Studio que usa a conexão e o conector para o Open Spark ElasticSearch.

Se seu trabalho for executado em uma Amazon Virtual Private Cloud (Amazon VPC), certifique-se de que a VPC esteja configurada corretamente. Para ter mais informações, consulte the section called "Configurar uma VPC para seu trabalho de ETL".

Para criar um trabalho que usa o Elasticsearch Spark Connector

1. Em AWS Glue Studio, escolha Connectors (Conectores).
2. Na lista Your connections (Suas conexões), selecione a conexão que você acabou de criar e escolha Create job (Criar trabalho).
3. No editor de trabalhos visual, escolha o nó de origem dos dados. À direita, na guia Data source properties - Connector (Propriedades da origem dos dados: conector), configure informações adicionais para o conector.

   a. Escolha Add schema (Adicionar esquema) e insira o esquema do conjunto de dados na origem dos dados. As conexões não usam tabelas armazenadas no Data Catalog, o que significa que AWS Glue Studio não conhece o esquema dos dados. Você deve fornecer manualmente essas informações de esquema. Para obter instruções sobre como usar o editor de esquemas, consulte the section called "Editar o esquema de um nó de transformação personalizada".

   b. Expanda Connection options (Opções de conexão).

   c. Escolha Add new option (Adicionar nova opção) e insira as informações necessárias para o conector que não foram inseridas no segredo da AWS:

      - es.nodes: https://<endpoint do domínio do OpenSearch>
      - es.port: 443
      - path: test
      - es.nodes.wan.only: true

      Para obter uma explicação sobre essas opções de conexão, consulte: https://www.elastic.co/guide/en/elasticsearch/hadoop/current/configuration.html.

4. Adicione um nó de destino ao gráfico.

Seu destino de dados pode ser o Amazon S3 ou pode usar informações de um AWS Glue Data Catalog ou um conector para gravar dados em um local diferente. Por exemplo, você pode usar uma tabela do Data Catalog para gravar em um banco de dados no Amazon RDS ou pode usar um conector como destino de dados para gravar em armazenamentos de dados que não são suportados nativamente no AWS Glue.

Se você escolher um conector para o destino dos dados, deverá escolher uma conexão criada para esse conector. Além disso, se exigido pelo provedor do conector, você deve adicionar opções para fornecer mais informações ao conector. Se você usar uma conexão que contenha informações para um segredo da AWS, então não será preciso fornecer a autenticação com o nome do usuário e senha nas opções de conexão.

5. Opcionalmente, adicione mais fontes de dados, e um ou mais nós de transformação, conforme descrito em the section called “Editar nós de transformação de dados gerenciados pelo AWS Glue”.
6. Configure as propriedades do trabalho conforme descrito em the section called “Modificar as propriedades do trabalho”, começando com a etapa 3, e salve o trabalho.

Próxima etapa

Etapa 6: Executar o trabalho

Etapa 6: Executar o trabalho

Depois de salvar o trabalho, você pode executá-lo para realizar as operações de ETL.

Para executar o trabalho que você criou para o AWS Glue Connector for Elasticsearch

1. Usando o console do AWS Glue Studio, na página do editor visual, escolha Run (Executar).
2. No banner de sucesso, escolha Run Details (Detalhes da execução), ou você pode escolher a opção Runs (Execuções) do editor visual para visualizar informações sobre a execução do trabalho.

Criando AWS Glue trabalhos com sessões interativas

Os engenheiros de dados podem criar trabalhos do AWS Glue de forma mais fácil e rápida do que antes usando sessões interativas no AWS Glue.

Tópicos

Visão geral das sessões interativas do AWS Glue

Com as sessões interativas do AWS Glue, você pode criar, testar e executar aplicações de análise e preparação de dados rapidamente. As sessões interativas fornecem uma interface programática e visual para criação e teste de scripts de extração, transformação e carregamento (ETL) para preparação de dados. As sessões interativas executam aplicações de análise do Apache Spark e fornecem acesso sob demanda a um ambiente do runtime remoto do Spark. O AWS Glue gerencia de forma transparente o Spark sem servidor para essas sessões interativas.

Sessões interativas são flexíveis, portanto você pode criar e testar aplicações a partir do ambiente de sua escolha. Você pode criar e trabalhar com sessões interativas por meio do AWS Command Line Interface e da API. Você pode usar cadernos compatíveis com o Jupyter para criar e testar visualmente seus scripts de caderno. As sessões interativas fornecem um kernel do Jupyter de código aberto que se integra em quase qualquer lugar que o Jupyter o faça, incluindo a integração com IDEs como PyCharm, IntelliJ e VS Code. Isso permite que você crie código em seu ambiente local e execute-o perfeitamente no backend de sessões interativas.

Usando a API de sessões interativas, os clientes podem executar programaticamente as aplicações que usam a análise do Apache Spark sem necessidade de gerenciar a infraestrutura do Spark. Você pode executar uma ou mais instruções do Spark em uma única sessão interativa.

As sessões interativas, portanto, fornecem uma maneira mais rápida, barata e mais flexível de criar e executar aplicações de análise e preparação de dados. Para aprender a usar sessões interativas, consulte a documentação nesta seção. Magic suportado pelo AWS Glue

Limitações

- Marcadores de trabalho não são compatíveis com sessões interativas.
- A criação de trabalhos de caderno usando a AWS Command Line Interface não tem suporte.

Conceitos básicos das sessões interativas do AWS Glue

Estas seções descrevem como executar sessões interativas do AWS Glue localmente.

Pré-requisitos para configurar sessões interativas localmente

Veja a seguir os pré-requisitos para instalar sessões interativas:

- As versões compatíveis do Python são 3.6 a 3.10+.
- Veja as seções abaixo para obter instruções para macOS/Linux e Windows.

Instalando o Jupyter e sessões AWS Glue interativas (kernels do Jupyter)

Use o seguinte para instalar o kernel localmente.

O comando, `install-glue-kernels`, instala o jupyter kernelspec para os kernels pyspark e spark e também instala logos no diretório correto.

```
pip3 install --upgrade jupyter boto3 aws-glue-sessions
```

```
install-glue-kernels
```

Execução do Jupyter

Conclua as etapas a seguir para executar o Jupyter Notebook.

1. Para executar o Jupyter Notebook, execute o comando a seguir.

```
jupyter notebook
```

2. Escolha New (Novo) e escolha um dos kernels do AWS Glue para começar a codificar no AWS Glue.

Configuração de credenciais e região da sessão

Instruções para macOS/Linux

As sessões interativas do AWS Glue exigem as mesmas permissões do IAM que trabalhos e endpoints de desenvolvimento do AWS Glue. Especifique a função usada com sessões interativas de uma das duas formas a seguir:

1. Com as mágicas `%iam_role` e `%region`
2. Com uma linha adicional em `~/.aws/config`

Configurar uma função de sessão com mágica

Na primeira célula, digite `%iam_role <YourGlueServiceRole>` na primeira célula executada.

Configurar uma função de sessão com **`~/.aws/config`**

AWS GlueA função de serviço para sessões interativas pode ser especificada no próprio notebook ou armazenada junto com a AWS CLI configuração. Se tiver uma função que você costuma usar com trabalhos do AWS Glue, essa será a função. Se não tiver um perfil que você usa para trabalhos do AWS Glue, siga as instruções neste guia Configurar permissões do IAM para o AWS Glue a fim de configurar um.

Para definir essa função como a função padrão para sessões interativas:

1. Em um editor de texto, abra `~/.aws/config`.
2. Procure o perfil que você usa para o AWS Glue. Se você não usar um perfil, use o perfil `[Default]`.

3. Adicione uma linha no perfil para a função que você pretende usar como `glue_role_arn=<AWSGlueServiceRole>`.
4. [Opcional]: se o seu perfil não tiver uma região padrão definida, é recomendável adicionar uma com `region=us-east-1`, substituindo `us-east-1` pela região desejada.
5. Salve a configuração.

Para ter mais informações, consulte Sessões interativas com o IAM.

Instruções para Windows

As sessões interativas do AWS Glue exigem as mesmas permissões do IAM que trabalhos e endpoints de desenvolvimento do AWS Glue. Especifique a função usada com sessões interativas de uma das duas formas a seguir:

1. Com as mágicas `%iam_role` e `%region`
2. Com uma linha adicional em `~/.aws/config`

Configurar uma função de sessão com mágica

Na primeira célula, digite `%iam_role <YourGlueServiceRole>` na primeira célula executada.

Configurar um perfil de sessão com o **`~/.aws/config`**

AWS GlueA função de serviço para sessões interativas pode ser especificada no próprio notebook ou armazenada junto com a AWS CLI configuração. Se tiver uma função que você costuma usar com trabalhos do AWS Glue, essa será a função. Se não tiver uma função que você usa para trabalhos do AWS Glue, siga as instruções no guia Configurar permissões do IAM para o AWS Glue a fim de configurar uma.

Para definir essa função como a função padrão para sessões interativas:

1. Em um editor de texto, abra `~/.aws/config`.
2. Procure o perfil que você usa para o AWS Glue. Se você não usar um perfil, use o perfil `[Default]`.
3. Adicione uma linha no perfil para a função que você pretende usar como `glue_role_arn=<AWSGlueServiceRole>`.
4. [Opcional]: se o seu perfil não tiver uma região padrão definida, é recomendável adicionar uma com `region=us-east-1`, substituindo `us-east-1` pela região desejada.

5. Salve a configuração.

Para ter mais informações, consulte Sessões interativas com o IAM.

Atualização com base na prévia das sessões interativas

O kernel foi atualizado com novos nomes quando foi lançado com a versão 0.27. Para limpar as versões de pré-visualização dos kernels, execute o seguinte em um terminal ou. PowerShell

> **Note**
>
> Se você fizer parte de qualquer outra prévia do AWS Glue que exija um modelo de serviço personalizado, remover o kernel removerá o modelo de serviço 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/
```

Usar sessões interativas com o SageMaker Studio

Sessões interativas do AWS Glue é um ambiente de runtime do Apache Spark sob demanda e sem servidor que cientistas e engenheiros de dados podem usar para criar, testar e executar aplicações de análise e preparação de dados rapidamente. É possível executar uma sessão interativa do AWS Glue iniciando um caderno do Amazon SageMaker Studio Classic.

Para obter mais informações, consulte Preparar dados usando sessões interativas do AWS Glue.

Uso de sessões interativas com o Microsoft Visual Studio Code

Pré-requisitos

- Instale as sessões interativas do AWS Glue e verifique se funcionam com o Jupyter Notebook.

- Baixe e instale o Visual Studio Code with Jupyter. Para mais detalhes, consulte Jupyter Notebook in VS Code (Jupyter Notebook no VS Code).

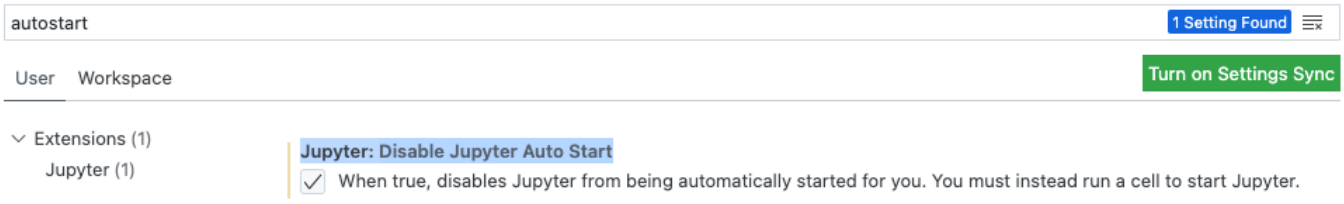
Para começar com sessões interativas com o VSCode

1. Desabilite a inicialização automática do Jupyter no VS Code.

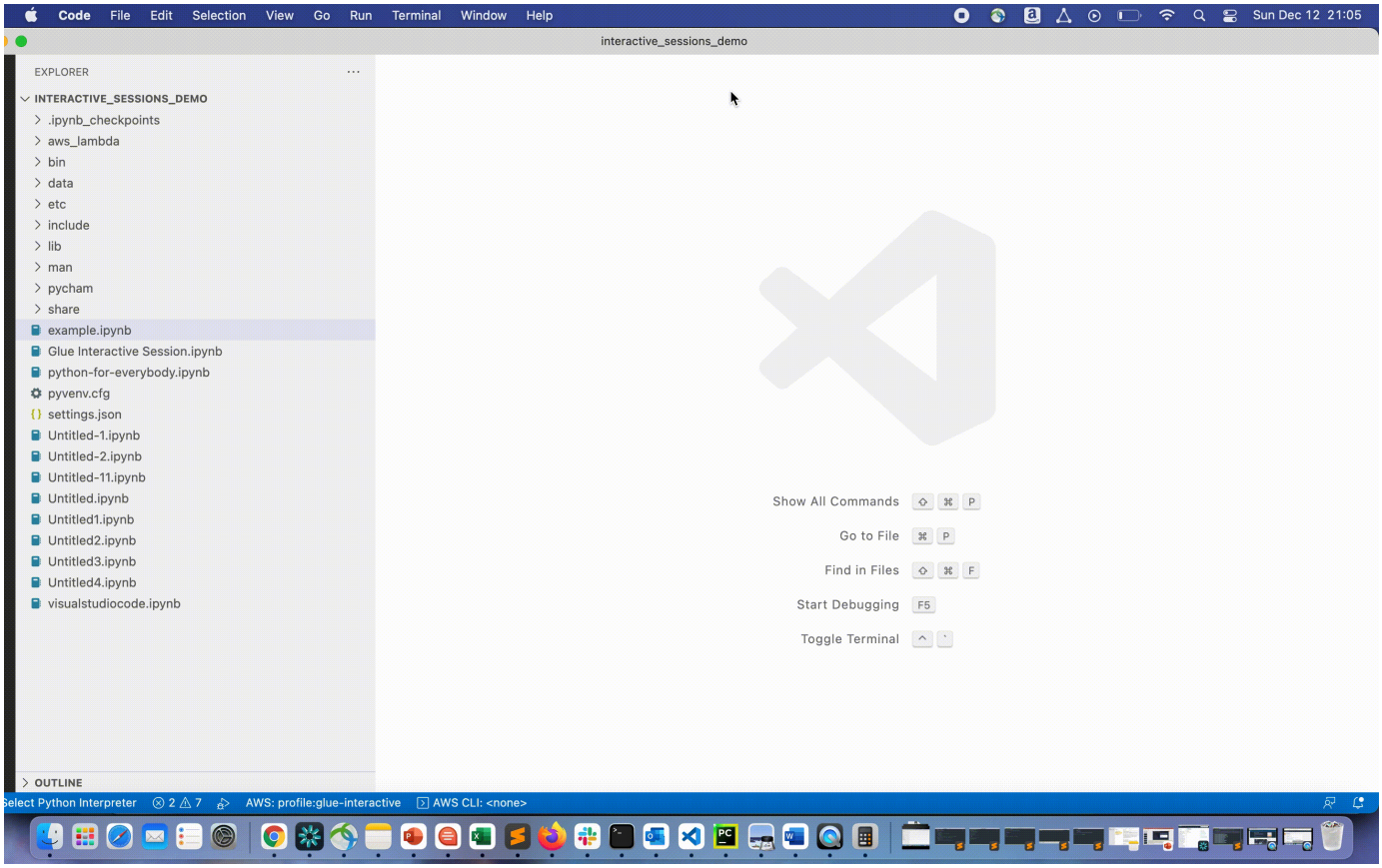
   No Visual Studio Code, os kernels do Jupyter serão iniciados automaticamente, o que impedirá que suas mágicas entrem em vigor, pois a sessão já terá sido iniciada. Para desativar o início automático no Windows, vá para Arquivo > Preferências > Extensões > Jupyter > clique com o botão direito do mouse em Jupyter e escolha Configurações de extensão.

   No MacOS, vá para Código > Configurações > Extensões > Jupyter > clique com o botão direito do mouse em Jupyter e escolha Configurações de extensão.

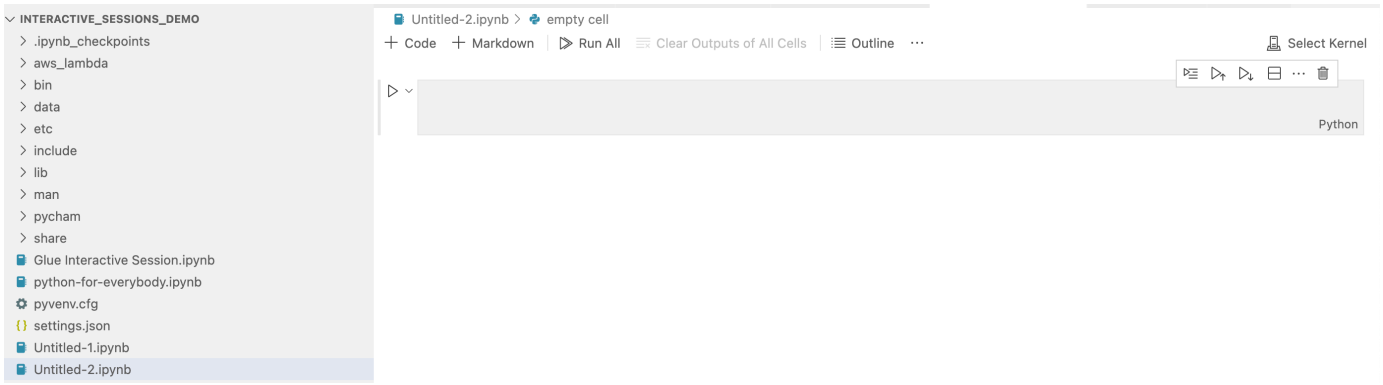
   Role para baixo até ver Jupyter: desativar o início automático do Jupyter. Marque a caixa “Quando verdadeiro, desativa a inicialização automática do Jupyter para você. Em vez disso, execute uma célula para iniciar o Jupyter."



2. Acesse File > New File > Save (Arquivo > Novo arquivo > Salvar) para salvar o arquivo atual com o nome de sua preferência com uma extensão `.ipynb` ou selecione jupyter em select a language (selecionar um idioma) e salve o arquivo.



3. Clique duas vezes no arquivo. O shell do Jupyter será exibido e um caderno será aberto.



4. No Windows, quando você cria um arquivo pela primeira vez, por padrão nenhum kernel é selecionado. Clique em Select Kernel (Selecionar kernel) e uma lista de kernels disponíveis será exibida. Escolha Glue PySpark.

   No MacOS, se você não encontrar o kernel do Glue PySpark, tente as seguintes etapas:

   1. Execute uma sessão local do Jupyter para obter o URL.

      Por exemplo, execute o comando a seguir para executar o caderno Jupyter.

```
jupyter notebook
```

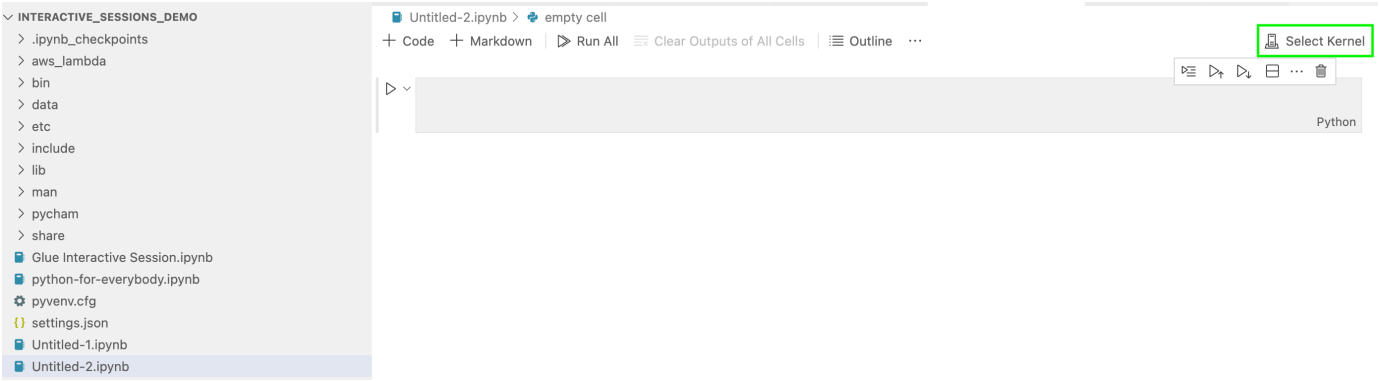
Quando o caderno for executado pela primeira vez, você verá uma URL parecida com `http://localhost:8888/?token=3398XXXXXXXXXXXXXXX`.

Copie o URL.

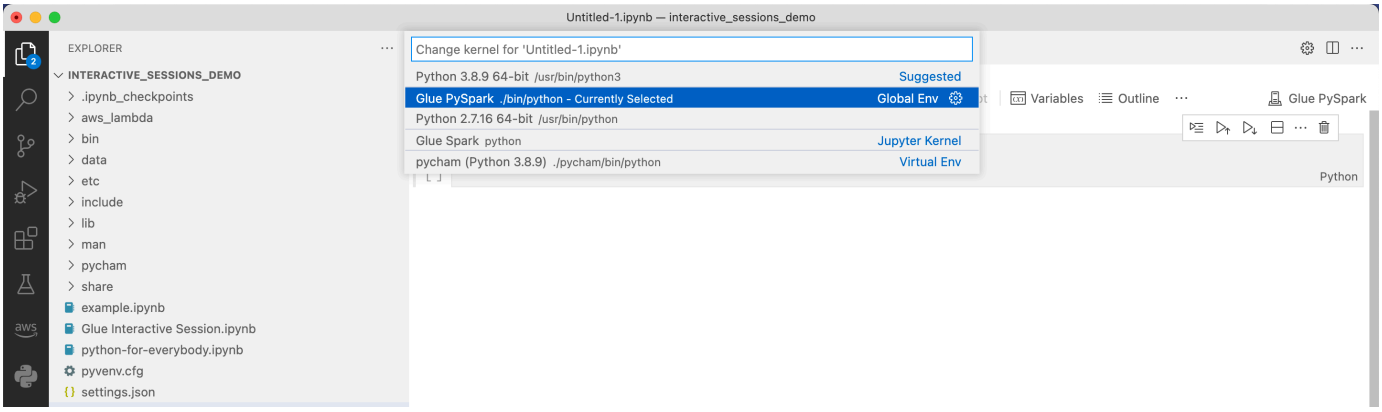
2. No VS Code, clique no kernel atual e Selecionar outro kernel..., e selecione Servidor Jupyter existente.... Cole o URL que você copiou da etapa acima.

   Se você receber uma mensagem de erro, consulte o wiki do VS Code Jupyter.

3. Se for bem-sucedido, isso definirá o kernel como Glue PySpark.



Selecione o kernel Glue PySpark ou Glue Spark (para Python e Scala, respectivamente).



Caso não encontre os kernels AWS Glue PySpark e AWS Glue Spark na lista suspensa, verifique se instalou o kernel AWS Glue na etapa acima ou se a configuração `python.defaultInterpreterPath` do Visual Studio Code está correta. Para obter mais informações, consulte a descrição da configuração python.defaultInterpreterPath.

5. Criar um AWS Glue Interactive Session. Prossiga para criar uma sessão da mesma maneira que você fez no caderno Jupyter. Especifique qualquer mágica na parte superior da primeira célula e execute uma declaração de código.

Configuração de sessões interativas do AWS Glue para cadernos do Jupyter e do AWS Glue Studio

Introdução às mágicas do Jupyter

As mágicas do Jupyter são comandos que podem ser executados no início de uma célula ou como um corpo inteiro de célula. As mágicas começam com `%` para mágicas de linha e `%%` para mágicas de células. Mágicas de linha, como `%region` e `%connections`, podem ser executadas com várias mágicas em uma célula ou com código incluído no corpo da célula, como no exemplo a seguir.

```
%region us-east-2
%connections my_rds_connection
dy_f = glue_context.create_dynamic_frame.from_catalog(database='rds_tables',
 table_name='sales_table')
```

As mágicas de célula devem usar toda a célula e podem fazer com que o comando englobe várias linhas. Veja um exemplo de `%%sql` abaixo.

```
%%sql
select * from rds_tables.sales_table
```

Mágicas compatíveis com sessões interativas do AWS Glue para Jupyter

Veja a seguir mágicas que você pode usar com sessões interativas do AWS Glue para cadernos do Jupyter.

Magics de sessões

Nome	Tipo	Descrição
`%help`	n/a	Retorna uma lista de descrições e tipos de entrada para todos os comandos magic.

Nome	Tipo	Descrição
`%profile`	String	Especifique um perfil em sua AWS configuração para usar como provedor de credenciais.
`%region`	String	Especifique o Região da AWS; no qual inicializar uma sessão. Padrão do `~/.aws/configure.` Exemplo: `%region us-west-1`
`%idle_timeout`	Int	O número de minutos de inatividade após o qual uma sessão atingirá o tempo limite após a execução de uma célula. O valor de tempo limite de inatividade padrão para sessões de ETL do Spark é o tempo limite padrão, 2.880 minutos (48 horas). Para outros tipos de sessão, consulte a documentação correspondente. Exemplo: `%idle_timeout 3000`
`%session_id`	n/a	Retorna o ID da sessão em execução.
`%session_id_prefix`	String	Definir uma string que precederá todos os IDs de sessão no formato[prefixo_ id_sessão]-[id_sessão]. Um UUID aleatório será gerado se um ID de sessão não for fornecido. Essa mágica não é compatível quando você executa um caderno Jupyter no AWS Glue Studio. Exemplo: `%session_id_prefix 001`
`%status`		Retornar o status da atual da sessão AWS Glue, incluindo sua duração, configura ção e usuário/função responsável pela execução.

Nome	Tipo	Descrição
`%stop_session`		Interrompe a sessão atual.
`%list_sessions`		Lista todas as sessões em execução atualmente por nome e ID.
`%session_type`	String	Define o tipo de sessão como Streaming, ETL ou Ray. Exemplo: `%session_type Streaming`
`%glue_version`	String	A versão do AWS Glue a ser usada pela sessão. Exemplo: `%glue_version 3.0`

Mágica para selecionar tipos de trabalho

Nome	Tipo	Descrição
`%streaming`	Cadeia de caracteres	Altera o tipo de sessão para transmissão do AWS Glue.
`%etl`	String	Altera o tipo de sessão para ETL do AWS Glue.
%glue_ray	String	Altera o tipo de sessão AWS Glue para Ray. Veja Magics apoiado pelas sessões interativas do AWS Glue Ray.

Mágicas de configuração do AWS Glue para Spark

A mágica `%%configure` é um dicionário formatado em json que consiste em todos os parâmetros de configuração para uma sessão. Cada parâmetro pode ser especificado aqui ou por meio de magics individuais.

Nome	Tipo	Descrição
`%%configure`	Dicionário	Especifica um dicionário formatado em JSON que consiste em todos os parâmetros de configuração para uma sessão. Cada parâmetro pode ser especificado aqui ou por meio de magics individuais. Para obter uma lista de parâmetros e exemplos de como usar o `%%configure` , consulte a tabela abaixo: Usar %%configure.
`%iam_role`	String	Especificar um ARN de função do IAM com o qual executar sua sessão. Padrão de ~/.aws/configure. Exemplo: `%iam_role AWSGlueServiceRole`
`%number_of_workers`	Int	O número de operadores de um worker_type definido que são alocados quando um trabalho é executado. `worker_type` também precisa estar definido. O `number_of_workers` padrão é 5. Exemplo: `%number_of_workers 2`
`%additional_python_modules`	Listar	Lista separada por vírgulas de módulos adicionais do Python para incluir no cluster (pode ser do Pypi ou do S3). Exemplo: `%additional_python_modules pandas, numpy` .
`%%tags`	String	Adiciona tags a uma sessão. Especifique as tags entre colchetes { }. Cada par de nomes de tag está entre parênteses (" ") e separado por uma vírgula (,).

Nome	Tipo	Descrição

		`%%tags {"billing":"Data-Platform", "team":"analytics"}`
		
		Use a mágica `%status` para ver as tags associadas à sessão.
		
		`%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']`

Nome	Tipo	Descrição

`%%assume_role`	Dicionário	Especifique um dicionário formatado em json ou uma string ARN do perfil do IAM para criar uma sessão para acesso entre contas.
		
		Exemplo com ARN:
		
		`%%assume_role`
		`{`
		`  'arn:aws:iam::XXXXXXXXXXXX:`
		`role/AWSGlueServiceRole'`
		`}`
		
		Exemplo com credenciais:
		
		` %%assume_role`
		`{{`
		`    "aws_access_key_id" =`
		` "XXXXXXXXXXXX",`
		`    "aws_secret_access_key" =`
		` "XXXXXXXXXXXX",`
		`    "aws_session_token" =`
		` "XXXXXXXXXXXX"`
		`}}`

%%configure argumentos da mágica da célula

A mágica `%%configure` é um dicionário formatado em json que consiste em todos os parâmetros de configuração para uma sessão. Cada parâmetro pode ser especificado aqui ou por meio de magics individuais. Veja abaixo exemplos de argumentos compatíveis com a mágica da célula `%%configure`. Use o prefixo `--` para os argumentos de execução especificados para o trabalho. Exemplo:

```
%%configure
{
   "--user-jars-first": "true",
```

```
   "--enable-glue-datacatalog": "false"
}
```

Para obter mais informações sobre parâmetros de trabalho, consulte Parâmetros de trabalho.

Configuração da sessão

Parâmetro	Tipo	Descrição

`max_retries`	Int	O número máximo de novas tentativas desse trabalho em caso de falha.
		`%%configure`
		`{`
		`  "max_retries": "0"`
		`}`

`max_concurrent_runs`	Int	O número máximo de execuções simultâneas permitidas para um trabalho.
		Exemplo:
		`%%configure`
		`{`
		`  "max_concurrent_runs": "3"`
		`}`

Parâmetros da sessão

Parâmetro	Tipo	Descrição
`--enable-spark-ui`	Booleano	Ative a interface de usuário do Spark para monitorar e depurar trabalhos de AWS Glue ETL. `%%configure`

Parâmetro	Tipo	Descrição

		<pre>{
		  "--enable-spark-ui": "true"
		}</pre>

`--spark-event-logs-path`	String	Especifica um caminho do Amazon S3. Quando usar o atributo de monitoramento da interface do usuário do Spark.
		
		Exemplo:
		
		<pre>%%configure
		{
		  "--spark-event-logs-path":
		 "s3://path/to/event/logs/"
		}</pre>

`--script_location`	String	Especifica o caminho do S3 para um script que executa um trabalho.
		
		Exemplo:
		
		<pre>%%configure
		{
		  "script_location": "s3://new-
		folder-here"
		}</pre>

Parâmetro	Tipo	Descrição

`--SECURITY_CONFIGURATION`	String	O nome de uma configuração AWS Glue de segurança
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--SECURITY_CONFIGURATION": "security-configuration-name",`
		`}`

`--job-language`	String	A linguagem de programação de script. Aceita um valor de 'scala' ou 'python'. O padrão é 'python'.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--job-language": "scala"`
		`}`

`--class`	String	A classe Scala que serve como ponto de entrada para o seu script Scala. O padrão é null.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--class": "className"`
		`}`

Parâmetro	Tipo	Descrição

`--user-jars-first`	Booleano	Prioriza os arquivos JAR extras do cliente no classpath. O padrão é null.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--user-jars-first": "true"`
		`}`

`--use-postgres-driver`	Booleano	Prioriza o driver JDBC Postgres no caminho da classe para evitar um conflito com o driver JDBC. Amazon Redshift O padrão é null.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--use-postgres-driver": "true"`
		`}`

Parâmetro	Tipo	Descrição

`--extra-files`	List(string)	Os caminhos do Amazon S3 para arquivos adicionais, como arquivos de configuração, que o AWS Glue copia para o diretório de trabalho do script antes de executá-lo.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--extra-files": "s3://path/to/additional/files/"`
		`}`

`--job-bookmark-option`	String	Controla o comportamento de um marcador de trabalho. Aceita um valor de job-bookmark-enable '', 'job-bookmark-disable' ou 'job-bookmark-pause'. O padrão é 'job-bookmark-disable'.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--job-bookmark-option": "job-bookmark-enable"`
		`}`

Parâmetro	Tipo	Descrição

`--TempDir`	String	Especifica um caminho do Amazon S3 para um bucket que pode ser usado como um diretório temporário para o trabalho. O padrão é null.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--TempDir": "s3://path/to/temp/dir"`
		`}`

`--enable-s3-parquet-optimized-committer`	Booleano	Habilita o committer otimizado para EMRFS otimizado para Amazon S3 a gravar dados Parquet no Amazon S3. O padrão é 'true'.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--enable-s3-parquet-optimized-committer": "false"`
		`}`

Parâmetro	Tipo	Descrição

`--enable-rename-algorithm-v2`	Booleano	Define a versão do algoritmo de renomeação do EMRFS como versão 2. O padrão é 'true'.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--enable-rename-algorithm-v2": "true"`
		`}`

`--enable-glue-datacatalog`	Booleano	Habilita o uso do catálogo de dados do AWS Glue como um metastore do Apache Spark Hive.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  --"enable-glue-datacatalog": "true"`
		`}`

`--enable-metrics`	Booleano	Habilita a coleta de métricas para criar perfis de trabalhos para a execução da tarefa. O padrão é 'false'.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--enable-metrics": "true"`
		`}`

Parâmetro	Tipo	Descrição

`--enable-continuous-cloudwatch-log`	Booleano	Habilita o registro em log contínuo em tempo real para trabalhos do AWS Glue. O padrão é 'false'.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--enable-continuous-cloudwatch-log": "true"`
		`}`

`--enable-continuous-log-filter`	Booleano	Especifica um filtro padrão ou nenhum filtro quando você cria ou edita um trabalho habilitado para registro em log contínuo. O padrão é 'true'.
		
		Exemplo:
		
		`%%configure`
		`{`
		`  "--enable-continuous-log-filter": "true"`
		`}`

Parâmetro	Tipo	Descrição
`--continuous-log-stream-prefix`	String	Especifica um prefixo de fluxo de Amazon CloudWatch log personalizado para um trabalho habilitado para registro contínuo. O padrão é null. Exemplo: `%%configure { "--continuous-log-stream-prefix": "prefix" }`
`--continuous-log-conversionPattern`	String	Especifica um padrão personalizado de conversão de logs para um trabalho habilitado para registro em log contínuo. O padrão é null. Exemplo: `%%configure { "--continuous-log-conversionPattern": "pattern" }`

Parâmetro	Tipo	Descrição

`--conf`	String	Controla os parâmetros de configuração do Spark. Destina-se a casos de uso avançados. Use `--conf` antes de cada parâmetro. Exemplo:
		`%%configure`
		`{`
		`"--conf": "spark.hadoop.hive`
		`.metastore.glue.catalogid=1`
		`23456789012 --conf hive.meta`
		`store.client.factory.class=`
		`com.amazonaws.glue.catalog.`
		`metastore.AWSGlueDataCatalo`
		`gHiveClientFactory --conf`
		`hive.metastore.schema.verif`
		`ication=false"`
		`}`

Mágica de trabalhos do Spark (ETL e streaming)

Nome	Tipo	Descrição
`%worker_type`	Cadeia de caracteres	Padrão, G.1X ou G.2X. `number_of_workers` também deve estar definido. O worker_type padrão é G.1X.

`%connections`	Listar	Especifica uma lista separada por vírgulas de conexões para usar na sessão.
		Exemplo:
		`%connections my_rds_connection`
		`dy_f =`
		`glue_context.create_dynamic`
		`_frame.from_catalog(databas`

Nome	Tipo	Descrição
		`e='rds_tables', table_nam e='sales_table')`
`%extra_py_files`	Listar	Lista separada por vírgulas de arquivos Python adicionais do Amazon S3.
`%extra_jars`	Listar	Lista separada por vírgulas de JARs adicionais para incluir no cluster.
`%spark_conf`	String	Especifique configurações personalizadas do Spark para sua sessão. Por exemplo, `%spark_conf spark.serializer=org.apache.spark.serializer.KryoSerializer` .

Mágicas para trabalhos do Ray

Nome	Tipo	Descrição
`%min_workers`	Int	O número mínimo de operadores alocados a um trabalho do Ray. Padrão: 1. Exemplo: `%min_workers 2`
`%object_memory_head`	Int	A porcentagem de memória livre no nó principal da instância após uma inicialização a quente. Mínimo: 0. Máximo: 100. Exemplo: `%object_memory_head 100`
%object_memory_worker	Int	A porcentagem de memória livre nos nós de processamento da instância após uma inicialização a quente. Mínimo: 0. Máximo: 100.

Nome	Tipo	Descrição
		Exemplo: `%object_memory_worker 100`

Magics de ação

Nome	Tipo	Descrição
`%%sql`	Cadeia de caracteres	Executar o código SQL. Todas as linhas após a mágica `%%sql` inicial serão transmitidas como parte do código SQL. Exemplo: `%%sql select * from rds_tables.sales_table`

`%matplot`	Figura Matplotlib	Visualize seus dados usando a biblioteca matplotlib.
		
		Exemplo:
		
		`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 Ploty	Visualize seus dados usando a biblioteca ploty. Exemplo:

Nome	Tipo	Descrição

		`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`

Nomeação de sessões

AWS Gluesessões interativas são AWS recursos e exigem um nome. Os nomes devem ser exclusivos para cada sessão e podem ser restringidos pelos administradores do IAM. Para ter mais informações, consulte Sessões interativas com o IAM. O kernel do Jupyter gera automaticamente nomes de sessão exclusivos para você. No entanto, é possível nomear as sessões manualmente de duas maneiras:

1. Usando o arquivo de AWS Command Line Interface configuração localizado em`~.aws/config`. Consulte Configurando o AWS Config com o. AWS Command Line Interface
2. Usar as mágicas `%session_id_prefix`. Consulte Mágicas compatíveis com sessões interativas do AWS Glue para Jupyter.

Um nome de sessão é gerado da seguinte forma:

- Quando o prefixo e o session_id forem fornecidos: o nome da sessão será {prefixo}-{UUID}.
- Quando nada for fornecido: o nome da sessão será {UUID}.

Prefixar nomes de sessão permite que você reconheça sua sessão ao listá-la no console AWS CLI ou.

Especificação de um perfil do IAM para sessões interativas

Você deve especificar uma função de AWS Identity and Access Management (IAM) para usar AWS Glue com o código ETL que você executa com sessões interativas.

A função requer as mesmas permissões do IAM que são necessárias para executar trabalhos do AWS Glue. Consulte Criar um perfil do IAM para o AWS Glue para obter mais informações sobre como criar uma função para trabalhos e sessões interativas do AWS Glue.

É possível especificar as funções do IAM de duas formas:

- Usando o arquivo de AWS Command Line Interface configuração localizado em `~.aws/config` (Recomendado). Para obter mais informações, consulte Configurar sessões com ~/.aws/config.

  > **Note**
  >
  > Quando a mágica `%profile` é usada, a configuração para `glue_iam_role` do respectivo perfil é respeitada.

- Uso da mágica %iam_role. Para ter mais informações, consulte Mágicas compatíveis com sessões interativas do AWS Glue para Jupyter.

Configurar sessões com perfis nomeados

AWS Glueas sessões interativas usam as mesmas credenciais do AWS Command Line Interface ou boto3, e as sessões interativas honram e funcionam com perfis nomeados, como os AWS CLI encontrados em (`~/.aws/config`Linux e macOS) ou (Windows). `%USERPROFILE%\.aws\config` Para obter mais informações, consulte Usar perfis nomeados.

As sessões interativas aproveitam os perfis nomeados permitindo que o perfil de serviço e o prefixo do ID da sessão do AWS Glue sejam especificados em um perfil. Para configurar uma função de perfil, adicione uma linha para a chave `iam_role` e/ou `session_id_prefix` ao seu perfil nomeado, conforme apresentado abaixo. O `session_id_prefix` não requer haspas. Por exemplo, se você quiser adicionar um `session_id_prefix`, insira o valor de `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>
```

Se tiver um método personalizado de geração de credenciais, você também poderá configurar seu perfil para usar o parâmetro `credential_process` em seu arquivo `~/.aws/config`. Por exemplo:

```
[profile developer]
region=us-east-1
credential_process = "/Users/Dave/generate_my_credentials.sh" --username helen
```

Você encontrará mais detalhes sobre a aquisição de credenciais por meio do parâmetro `credential_process` em: Credenciais de fornecimento com um processo externo.

Se não houver uma região ou `iam_role` definidas no perfil que você está usando, será necessário especificá-las usando as mágicas `%region` e `%iam_role` na primeira célula que você executar.

Introdução às sessões interativas do AWS Glue for Ray (versão prévia)

Warning

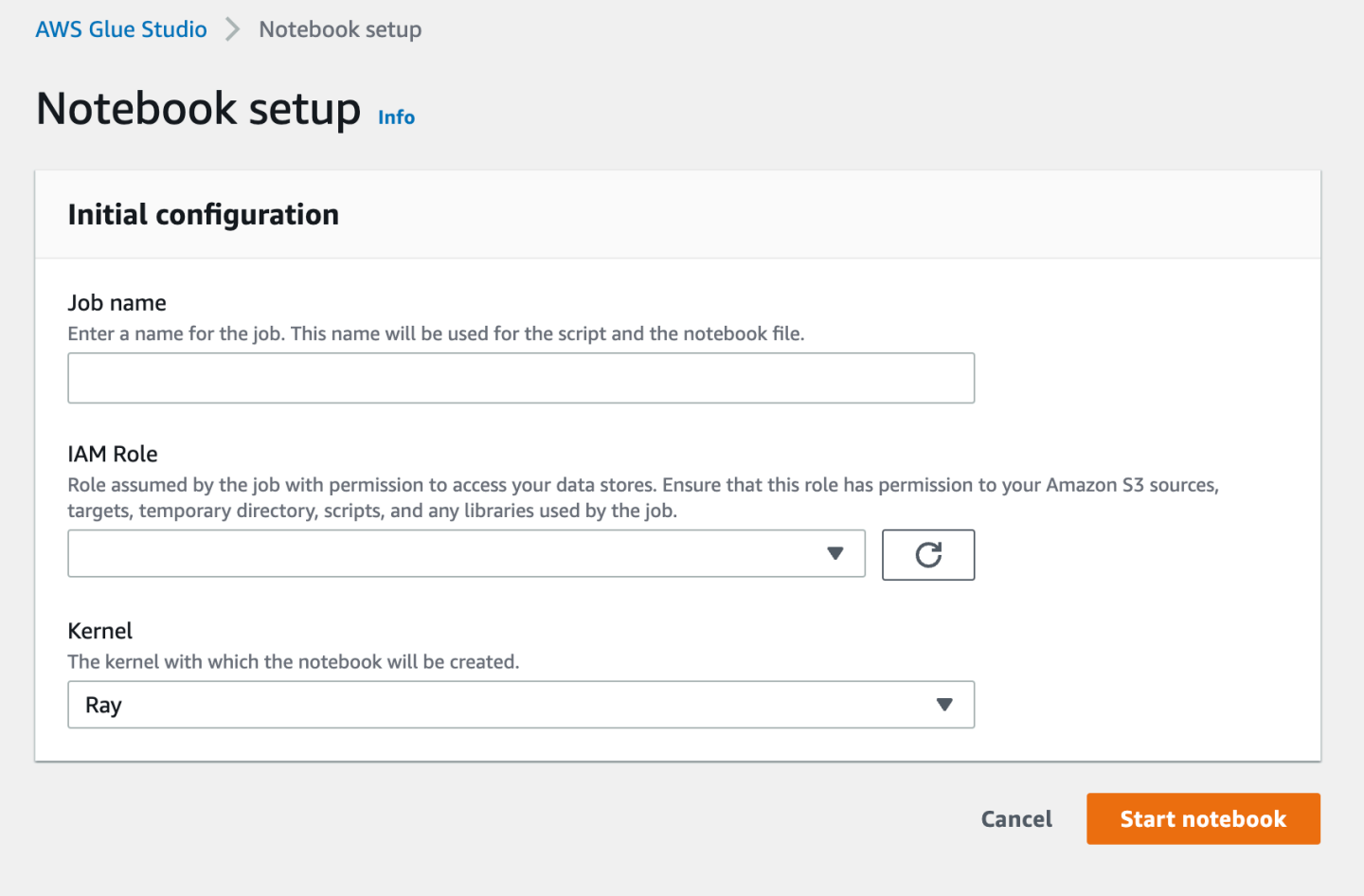
A prévia das sessões interativas do AWS Glue for Ray terminou em 30 de abril de 2024. Você não poderá mais criar novas sessões interativas no AWS Glue for Ray.

Note

AWS Glue for Ray está disponível no Leste dos EUA (Norte da Virgínia), Leste dos EUA (Ohio), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio) e Europa (Irlanda).

Sessões interativas do Ray no AWS Glue Studio console

Na página Trabalhos no AWS Glue Studio Console, selecione a opção Jupyter Notebook existente. Isso abrirá uma página de Notebook setup (Configuração do caderno), na qual você poderá selecionar o Kernel. Selecione o kernel do Ray para iniciar uma sessão interativa do Ray. Para obter mais informações sobre sessões interativa e sobre como elas são usadas, consulte the section called “Conceitos básicos das sessões interativas do AWS Glue”.



Sessões interativas de Ray usando o kernel do Jupyter

Para usar o Ray Kernel fora do AWS Glue Studio console, você precisará instalar o `aws-glue-sessions` pacote, que publicamos no PyPI. Para obter mais informações sobre como usar o pacote do kernel, consulte a documentação do the section called “Conceitos básicos das sessões interativas do AWS Glue”.

Para atualizar ou instalar o kernel, execute `pip install --upgrade aws-glue-sessions`. Você precisará da versão .37+ para usar o kernel do Ray.

As sessões interativas do Ray têm acesso às mesmas bibliotecas e versões do Ray que os trabalhos do Ray. Na versão prévia, todas as sessões interativas do Ray usarão o Ray 2.4.0.

Valores padrão de tempo limite de sessão interativa do Ray

- Tempo limite (por sessão) padrão: 8 horas.
- Tempo limite de natividade padrão: 1 hora.

Mágicas compatíveis com sessões interativas do AWS Glue Ray

As mágicas do kernel do Jupyter do AWS Glue quando ele é a tecnologia usada nas sessões interativas do Ray são semelhantes às das sessões do Spark. Para referência, consulte the section called " Configuração de sessões interativas do AWS Glue para cadernos do Jupyter e do AWS Glue Studio".

Mágicas de sessões

A mágica das sessões são basicamente as mesmas de antes da versão pré-lançamento do AWS Glue for Ray. Para obter mais informações sobre as mágicas de sessão fora desta versão pré-lançamento, consulte the section called "Mágicas compatíveis com sessões interativas do AWS Glue para Jupyter". Apresentamos uma nova mágica para definir o tipo de sessão como AWS Glue for Ray.

Nome	Tipo	Descrição
`%glue_ray`	Cadeia de caracteres	Altera o tipo de sessão para AWS Glue for Ray.

Magics de configuração do AWS Glue

As mágicas para configurar o AWS Glue em uma sessão interativa podem ser diferentes entre os tipos de sessão. Atualmente, damos suporte apensa a esse subconjunto de mágicas existentes ao usar AWS Glue for Ray.

> **Warning**
>
> Problema conhecido: **`additional_python_modules`**

Na prévia das sessões interativas do Ray, o uso da mágica `additional_python_modules` causará problemas ao salvar seu caderno. Para configurar módulos python para sessões de Ray, use a mágica `%%configure` para definir o parâmetro `pip-install` definido emthe section called “Parâmetros de trabalho do Ray”.

Nome	Tipo	Descrição
`%%configure`	Dicionário	Especifica um dicionário formatado em JSON que consiste em todos os parâmetros de configuração para uma sessão. Cada parâmetro pode ser especificado aqui ou por meio de magics individuais.
`%iam_role`	String	Especificar um ARN de função do IAM com o qual executar sua sessão. Padrão com base em ~/.aws/configure
`%number_of_workers`	int	O número de operadores de um worker_type definido que são alocados quando um trabalho é executado. `worker_type` também precisa estar definido.
`%worker_type`	String	Na prévia do AWS Glue for Ray, o único tipo de processador compatível é o Z.2X.
`%additional_python_modules`	Lista	Lista separada por vírgulas de módulos Python adicionais para incluir em seu cluster (pode ser do Pypi ou S3).

Magics de ação

As sessões do AWS Glue Ray não são compatíveis com nenhuma mágica de ação.

Sessões interativas com o IAM

Essas seções descrevem as considerações de segurança para sessões interativas do AWS Glue.

Tópicos

Entidades principais do IAM usadas com sessões interativas

Você usa duas entidades principais do IAM com sessões interativas do AWS Glue

- Client principal (Entidade principal do cliente): a entidade principal do cliente (um usuário ou perfil) autoriza operações de API para sessões interativas de um cliente do AWS Glue configurado com as credenciais baseadas na identidade da entidade principal. Por exemplo, poderia ser um perfil do IAM que você normalmente usa para acessar o console do AWS Glue. Isso também pode ser uma função atribuída a um usuário no IAM cujas credenciais são usadas para o AWS Command Line Interface, ou a um AWS Glue cliente usado pelas sessões interativas do kernel Jupyter.
- Perfil de runtime: o perfil de runtime é um perfil do IAM que a entidade principal do cliente repassa para as operações de API de sessões interativas. O AWS Glue usa esse perfil para executar instruções na sessão. Por exemplo, essa função pode ser a usada para execução de trabalhos de ETL do AWS Glue.

  Para ter mais informações, consulte Configuração de uma função do runtime.

Configuração de uma entidade principal do cliente

Você deve anexar uma política de identidade à entidade principal do cliente para permitir que ela chame a API de sessões interativas. Esse perfil deve ter acesso de `iam:PassRole` ao perfil de execução que você repassaria para a API de sessões interativas, como `CreateSession`. Por exemplo, você pode anexar a política AWSGlueConsoleFullAccessgerenciada a uma função do IAM que permite que os usuários em sua conta com a política anexada acessem todas as sessões criadas em sua conta (como declaração de tempo de execução ou declaração de cancelamento).

Se você quiser proteger sua sessão e torná-la privada somente para determinadas funções do IAM, como aquelas associadas ao usuário que criou a sessão, você pode usar o Controle de Autorização Baseado em Tags da AWS Glue Interactive Session, chamado TagOnCreate. Para obter mais

informações, veja Torne sua sessão privada com TagOnCreate como uma política gerenciada com escopo reduzido baseada na etiqueta do proprietário pode tornar sua sessão privada com. TagOnCreate Para obter mais informações sobre políticas baseadas em identidade, consulte Políticas baseadas em identidade para o AWS Glue.

Configuração de uma função do runtime

Você deve passar uma função do IAM para a operação da CreateSession API AWS Glue para permitir assumir e executar instruções em sessões interativas. A função deve ter as mesmas permissões do IAM que as necessárias para execução de um trabalho típico do AWS Glue. Por exemplo, você pode criar uma função de serviço usando a AWSGlueServiceRolepolítica que AWS Glue permite chamar AWS serviços em seu nome. Se você usar o console do AWS Glue, ele criará automaticamente uma função de serviço em seu nome ou usará uma existente. Você também pode criar sua própria função do IAM e anexar sua própria política do IAM para permitir permissões semelhantes.

Se você quiser proteger sua sessão e torná-la privada somente para o usuário que criou a sessão, você pode usar o Controle de Autorização Baseado em Tags da Sessão AWS Glue Interativa, chamado TagOnCreate. Para obter mais informações, veja Torne sua sessão privada com TagOnCreate como uma política gerenciada com escopo reduzido baseada na etiqueta do proprietário pode tornar sua sessão privada com. TagOnCreate Para obter mais informações sobre políticas baseadas em identidade, consulte Políticas baseadas em identidade para Glue AWS. Se você estiver criando a função de execução sozinho a partir do console do IAM e quiser tornar seu serviço privado com o TagOnCreate recurso, siga as etapas abaixo.

1. Crie uma função do IAM com o tipo de função definido como `Glue`.
2. Anexe esta política AWS Glue gerenciada: AwsGlueSessionUserRestrictedServiceRole
3. Prefixe o nome da função com o nome AwsGlueSessionUserRestrictedServiceRoleda política. Por exemplo, você pode criar uma função com o nome AwsGlueSessionUserRestrictedServiceRole-myrole e anexar uma política AWS Glue gerenciada. AwsGlueSessionUserRestrictedServiceRole
4. Anexe uma política de confiança como a seguinte para permitir que o AWS Glue assuma a função:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "glue.amazonaws.com"
        ]
      },
      "Action": [
        "sts:AssumeRole"
      ]
    }
  ]
}
```

Para um kernel Jupyter de sessões interativas, você pode especificar a `iam_role` chave em seu perfil. AWS Command Line Interface Para obter mais informações, consulte Configurar sessões com ~/.aws/config. Se você estiver interagindo com sessões interativas usando um bloco de anotações do AWS Glue, poderá passar o perfil de execução no magic `%iam_role`, na primeira célula que executar.

Torne sua sessão privada com TagOnCreate

As sessões interativas do AWS Glue são compatíveis com marcação e Tag Based Authorization (TBAC – Autorização baseada em etiquetas) para sessões interativas como um recurso nomeado. Além do uso do TBAC TagResource e das UntagResource APIs, as sessões AWS Glue interativas oferecem suporte ao TagOnCreate recurso de “marcar” uma sessão com uma determinada tag somente durante a criação da sessão com operação. CreateSession Isso também significa que essas tags serão removidas em DeleteSession, aka UntagOnDelete.

TagOnCreate oferece um poderoso mecanismo de segurança para tornar sua sessão privada para o criador da sessão. Por exemplo, você pode anexar uma política do IAM com “proprietário” RequestTag e valor de $ {aws:userId} a um cliente principal (como um usuário) para permitir a criação de uma sessão somente se uma tag “proprietário” com o valor correspondente do nome de usuário do chamador for fornecida como tag de ID de usuário na solicitação. CreateSession Essa política permite que as sessões interativas do AWS Glue criem um recurso de sessão e marquem a sessão com a etiqueta userId somente durante o tempo de criação da sessão. Além disso, você pode restringir o acesso (como instruções em execução) à sua sessão somente para o criador (também conhecido como tag de proprietário com valor $ {aws:userId}) da sessão anexando uma política do IAM com “proprietário” ResourceTag à função de execução que você passou durante a execução. CreateSession

Para facilitar o uso do TagOnCreate recurso de tornar uma sessão privada para o criador da sessão, AWS Glue fornece políticas gerenciadas e funções de serviço especializadas.

Se você quiser criar uma sessão AWS Glue interativa usando um AssumeRole principal do IAM (ou seja, usando uma credencial vendida assumindo uma função do IAM) e quiser tornar a sessão privada para o criador, use políticas semelhantes às e, respectivamente. AWSGlueSessionUserRestrictedNotebookPolicyAWSGlueSessionUserRestrictedNotebookServiceRole Essas políticas permitem AWS Glue usar $ {aws:PrincipalTag} para extrair o valor da tag do proprietário. Isso exige que você passe uma tag de ID de usuário com o valor $ {aws:userId}, como na credencial de assumir a função. SessionTag Consulte Etiquetas de sessão de ID. Se você estiver usando uma instância do Amazon EC2 com um perfil de instância vendendo a credencial e quiser criar uma sessão ou interagir com a sessão de dentro da instância do Amazon EC2, precisará passar uma tag de ID de usuário com o valor $ {aws:userId} como na credencial assume a função. SessionTag

Por exemplo, se você estiver criando uma sessão usando uma credencial AssumeRole principal do IAM e quiser tornar seu serviço privado com o TagOnCreate recurso, siga as etapas abaixo.

1. Crie você mesmo uma função de runtime no console do IAM. Anexe essa política AWS Glue gerenciada AwsGlueSessionUserRestrictedNotebookServiceRolee prefixe o nome da função com o nome da política. AwsGlueSessionUserRestrictedNotebookServiceRole Por exemplo, você pode criar uma função com o nome AwsGlueSessionUserRestrictedNotebookServiceRole-myrole e anexar uma política AWS Glue gerenciada. AwsGlueSessionUserRestrictedNotebookServiceRole
2. Anexe uma política de confiança como a exibida abaixo para permitir que o AWS Glue assuma a função acima.

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

```
    }
   ]
}
```

3. Crie outra função nomeada com um prefixo AwsGlueSessionUserRestrictedNotebookPolicye anexe a política AWS Glue gerenciada AwsGlueSessionUserRestrictedNotebookPolicypara tornar a sessão privada. Além da política gerenciada, anexe a seguinte política embutida para permitir iam: PassRole à função que você criou na etapa 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. Anexe uma política de confiança como a seguinte ao IAM acima para que o AWS Glue assuma a função.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": [
```

```
                "glue.amazonaws.com"
            ]
        },
        "Action": [
            "sts:AssumeRole",
            "sts:TagSession"
        ]
    }]
}
```

> **Note**
>
> Opcionalmente, você pode usar uma única função (por exemplo, função de notebook) e anexar as políticas gerenciadas acima AwsGlueSessionUserRestrictedNotebookServiceRolee. AwsGlueSessionUserRestrictedNotebookPolicy Anexe também a política em linha adicional para permitir `iam:passrole` da sua função para o AWS Glue. Por fim, anexe a política de confiança acima para permitir `sts:AssumeRole` e `sts:TagSession`.

AWSGlueSessionUserRestrictedNotebookPolicy

O AWSGlueSessionUserRestrictedNotebookPolicy fornece acesso para criar uma sessão AWS Glue interativa a partir de um notebook somente se uma chave de tag “proprietário” e um valor AWS corresponderem ao ID de usuário do principal (usuário ou função). Para mais informações, consulte Onde você pode usar variáveis de política. Essa política é anexada à entidade principal (usuário ou perfil) que cria cadernos de sessões interativas do AWS Glue no AWS Glue Studio. Essa política também permite acesso suficiente ao AWS Glue Studio notebook para interagir com os recursos da Sessão AWS Glue Studio Interativa que são criados com o valor da tag “proprietário” correspondente ao ID de AWS usuário do principal. Essa política nega permissão para alterar ou remover etiquetas “owner” de um recurso de sessão do AWS Glue após a criação da sessão.

AWSGlueSessionUserRestrictedNotebookServiceRole

O AWSGlueSessionUserRestrictedNotebookServiceRolefornece acesso suficiente ao AWS Glue Studio notebook para interagir com os recursos da Sessão AWS Glue Interativa que são criados com o valor da tag “owner” correspondente ao ID do AWS usuário principal (usuário ou função) do criador do notebook. Para mais informações, consulte Onde você pode usar variáveis de política. Essa

política de função de serviço é anexada à função que é passada como mágica para um notebook ou passada como função de execução para a CreateSession API. Essa política também permite criar uma sessão AWS Glue interativa a partir de um notebook somente se a tag, a chave “proprietário” e o valor corresponderem ao ID de AWS usuário do principal. Essa política nega permissão para alterar ou remover etiquetas “owner” de um recurso de sessão do AWS Glue após a criação da sessão. Essa política também inclui permissões para gravação e leitura de buckets do Amazon S3, gravação de CloudWatch registros, criação e exclusão de tags para recursos do Amazon EC2 usados pelo. AWS Glue

Tornar sua sessão privada com políticas de usuário

Você pode AWSGlueSessionUserRestrictedPolicyanexar as duas funções do IAM associadas a cada um dos usuários em sua conta para impedir que eles criem uma sessão somente com uma tag de proprietário com um valor correspondente a seus próprios $ {aws:userId}. Em vez de usar o AWSGlueSessionUserRestrictedNotebookPolicye, AWSGlueSessionUserRestrictedNotebookServiceRolevocê precisa usar políticas semelhantes às AWSGlueSessionUserRestrictedPolicye AWSGlueSessionUserRestrictedServiceRole, respectivamente. Para mais informações, consulte Usar políticas baseadas em identidade. Essa política limita o escopo do acesso a uma sessão apenas ao criador, a ${aws:userId} do usuário que criou a sessão com a tag "owner" com seu próprio ${aws:userId}. Se você mesmo criou a função de execução usando o console do IAM seguindo as etapas emConfiguração de uma função do runtime, além de anexar a política AwsGlueSessionUserRestrictedPolicygerenciada, anexe a seguinte política embutida a cada um dos usuários em sua conta `iam:PassRole` para permitir a função de execução que você criou 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

O AWSGlueSessionUserRestrictedPolicyfornece acesso para criar uma sessão AWS Glue interativa usando a CreateSession API somente se uma chave de tag “proprietário” e um valor correspondente ao ID de AWS usuário forem fornecidos. Essa política de identidade é anexada ao usuário que invoca a CreateSession API. Essa política também permite interagir com os recursos da Sessão AWS Glue Interativa que foram criados com uma tag de “proprietário” e um valor que correspondam ao ID de AWS usuário. Essa política nega permissão para alterar ou remover etiquetas “owner” de um recurso de sessão do AWS Glue após a criação da sessão.

AWSGlueSessionUserRestrictedServiceRole

O AWSGlueSessionUserRestrictedServiceRolefornece acesso total a todos os AWS Glue recursos, exceto às sessões, e permite que os usuários criem e usem somente as sessões interativas associadas ao usuário. Essa política também inclui outras permissões necessárias AWS Glue para gerenciar os recursos do Glue em outros AWS serviços. A política também permite adicionar tags a AWS Glue recursos em outros AWS serviços.

Considerações sobre políticas do IAM

Sessões interativas são recursos do IAM no AWS Glue. Como elas são recursos do IAM, o acesso e a interação a uma sessão são regidos pelas políticas do IAM. Com base nas políticas do IAM anexadas a uma entidade principal do cliente ou função de execução configurada por um administrador, uma entidade principal do cliente (usuário ou função) poderá criar novas sessões e interagir com suas próprias sessões e outras sessões.

Se um administrador tiver anexado uma política do IAM, como AWSGlueConsoleFullAccess ou AWSGlueServiceRole que permita acesso a todos os AWS Glue recursos dessa conta, o principal do cliente poderá colaborar entre si. Por exemplo, se as políticas permitirem, um usuário do IAM poderá interagir com as sessões criadas por outros usuários.

Se você quiser configurar uma política ajustada às suas necessidades específicas, consulte a documentação do IAM sobre configuração de recursos para uma política. Por exemplo, para isolar sessões que pertencem a um usuário, você pode usar o TagOnCreate recurso suportado pelas sessões AWS Glue interativas. Consulte Torne sua sessão privada com TagOnCreate .

As sessões interativas são compatíveis com a limitação de criação de sessões com base em determinadas condições da VPC. Consulte Controlar políticas que controlam configurações usando chaves de condição.

Converter um script ou caderno em um trabalho do AWS Glue

Há duas formas de converter um script ou caderno em um trabalho do AWS Glue:

- Usar nbconvert para converter seu arquivo de documento de caderno `.ipynb` do Jupyter em um arquivo `.py`. Para mais informações, consulte nbconvert: Convert Notebooks to other formats (nbconvert: converter cadernos para outros formatos).
- Carregue o arquivo no AWS Glue Studio Notebooks.
  - No console do AWS Glue Studio, escolha Jobs (Trabalhos) no menu de navegação.
  - Na seção Create job (Criar trabalho), escolha Jupyter Notebook.
  - Na seção Options (Opções), escolha Upload and edit an existing notebook (Carregar e editar um caderno existente).
  - Selecione Choose file (Escolher arquivo) para carregar um arquivo `.ipynb`.

Sessões interativas do AWS Glue para transmissão

Alterar o tipo de sessão de transmissão

Use a mágica de configuração das sessões interativas do AWS Glue, `%streaming`, para definir o trabalho que você está executando e inicializar uma sessão interativa de transmissão.

Amostrar o fluxo de entrada para desenvolvimento interativo

Uma ferramenta que desenvolvemos para ajudar a aprimorar a experiência AWS Glue interativa em sessões interativas é a adição de um novo método `GlueContext` para obter um instantâneo de um stream em uma estática DynamicFrame. `GlueContext`permite que você inspecione, interaja e implemente seu fluxo de trabalho.

Com a instância de classe `GlueContext`, você poderá localizar o método `getSampleStreamingDynamicFrame`. Os argumentos necessários para esse método são:

- `dataFrame`: O streaming do Spark DataFrame

- `options`: veja as opções disponíveis abaixo

As opções disponíveis incluem:

- windowSize: também chamado de Microbatch Duration (Duração de microlote). Esse parâmetro determinará quanto tempo uma consulta de transmissão aguardará após o acionamento do lote anterior. Esse valor de parâmetro deve ser inferior a `pollingTimeInMs`.
- pollingTimeInMs: O tempo total em que o método será executado. Ele acionará pelo menos um microlote para obter registros de amostra do fluxo de entrada.
- recordPollingLimit: esse parâmetro ajuda a limitar o número total de registros que você pesquisará no stream.
- (Opcional) Você também pode usar `writeStreamFunction` para aplicar essa função personalizada a cada função de amostragem de registro. Veja abaixo exemplos em Scala e Python.

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**
>
> Quando o `DynFrame` amostrado estiver vazio, isso pode acontecer por alguns motivos:
>
> - A fonte de transmissão está definida como “Latest” (Mais recente), e nenhum novo dado foi ingerido durante o período de amostragem.
> - O tempo de sondagem não é suficiente para processar os registros ingeridos. Os dados não serão exibidos, a menos que o lote inteiro tenha sido processado.

Executar aplicativos de transmissão em sessões interativas

Em sessões interativas do AWS Glue, você pode executar o aplicativo de transmissão do AWS Glue assim como criaria um aplicativo de transmissão no console do AWS Glue. Como as sessões interativas são baseadas em sessão, encontrar exceções no runtime não faz com que a sessão seja interrompida. Agora temos o benefício adicional de desenvolver sua função em lote iterativamente. Por exemplo:

```
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 =
 {**})
```

No exemplo acima, incluímos um uso inválido de um método e, ao contrário dos trabalhos normais do AWS Glue que sairão de toda a aplicação, o contexto de codificação e as definições do usuário são totalmente preservados e a sessão permanece operacional. Não há necessidade de fazer o bootstrap de um novo cluster e executar novamente toda a transformação anterior. Isso permite que você mantenha o foco em iterar rapidamente suas implementações de função em lote para obter resultados desejáveis.

É importante observar que o Interactive Session avalia cada instrução de maneira bloqueadora para que a sessão execute apenas uma instrução por vez. Como as consultas de transmissão são contínuas e infinitas, as sessões com consultas de transmissão ativa não poderão processar instruções de acompanhamento a menos que sejam interrompidas. Você pode emitir o comando de interrupção diretamente do Jupyter Notebook e nosso kernel processará o cancelamento para você.

Considere como exemplo a seguinte sequência de instruções que estão aguardando a execução:

```
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
```

Desenvolver e testar scripts de trabalho do AWS Glue localmente

Ao desenvolver e testar seus scripts de trabalho do AWS Glue para Spark, existem várias opções disponíveis:

- Console do AWS Glue Studio
  - Editor visual
  - Editor de scripts
  - Bloco de anotações do AWS Glue Studio
- Sessões interativas
  - Bloco de anotações Jupyter
- Docker image (Imagem do Docker)
  - Desenvolvimento local
  - Desenvolvimento remoto
- Biblioteca ETL do AWS Glue Studio
  - Desenvolvimento local

É possível escolher qualquer uma das opções acima dependendo das suas necessidades.

Se preferir nenhum código ou tiver menos experiência com código, o editor visual do AWS Glue Studio é uma boa opção.

Se preferir uma experiência interativa de caderno, o caderno do AWS Glue Studio é uma boa opção. Para obter mais informações, consulte Usar cadernos com o AWS Glue Studio e o AWS Glue. Se quiser usar seu próprio ambiente local, sessões interativas são uma boa opção. Para obter mais informações, consulte Usar Interactive Sessions com o AWS Glue.

Se você prefere a experiência de desenvolvimento local/remoto, a imagem do Docker é uma boa opção. Isso ajuda a desenvolver e testar scripts de trabalho do AWS Glue para Spark em qualquer lugar de sua preferência sem gerar custos com a AWS Glue.

Se preferir o desenvolvimento local sem o Docker, a instalação do diretório da biblioteca ETL do AWS Glue localmente é uma boa opção.

Desenvolver usando o AWS Glue Studio

O editor visual do AWS Glue Studio é uma interface gráfica que facilita a criação, a execução e o monitoramento de trabalhos do tipo extrair, transformar e carregar (ETL) no AWS Glue. Você pode compor visualmente fluxos de trabalho de transformação de dados e executá-los perfeitamente no mecanismo de ETL sem servidor baseado no Apache Spark do AWS Glue. Você pode inspecionar os resultados do esquema e dos dados em cada etapa do trabalho. Para mais informações, consulte o Guia do usuário do AWS Glue Studio.

Desenvolvimento com sessões interativas

Sessões interativas permitem criar e testar aplicações no ambiente de sua escolha. Para obter mais informações, consulte Usar Interactive Sessions com o AWS Glue.

Desenvolvimento com uma imagem do Docker

> **Note**
>
> As instruções nesta seção não foram testadas em sistemas operacionais Microsoft Windows. Para desenvolvimento local e testes em plataformas Windows, consulte o blog Desenvolvimento de um pipeline de ETL do AWS Glue localmente sem uma conta da AWS

Para uma plataforma de dados pronta para produção, o processo de desenvolvimento e o pipeline de CI/CD para trabalhos do AWS Glue é um tópico chave. Você pode desenvolver e testar com flexibilidade trabalhos do AWS Glue em um contêiner do Docker. O AWS Glue hospeda imagens do

Docker no Docker Hub para configurar seu ambiente de desenvolvimento com utilitários adicionais. Você pode usar seu IDE, caderno ou REPL preferido com a biblioteca ETL do AWS Glue. Este tópico descreve como desenvolver e testar trabalhos do AWS Glue versão 4.0 em um contêiner do Docker usando uma imagem do Docker.

As seguintes imagens do Docker estão disponíveis para o AWS Glue no Docker Hub.

- Para o AWS Glue versão 4.0: `amazon/aws-glue-libs:glue_libs_4.0.0_image_01`
- Para o AWS Glue versão 3.0: `amazon/aws-glue-libs:glue_libs_3.0.0_image_01`
- Para o AWS Glue versão 2.0: `amazon/aws-glue-libs:glue_libs_2.0.0_image_01`

Essas imagens são para ambientes x86_64. É recomendável que você teste essa arquitetura. No entanto, talvez seja possível retrabalhar uma solução de desenvolvimento local em imagens básicas não suportadas.

Esse exemplo descreve o uso do `amazon/aws-glue-libs:glue_libs_4.0.0_image_01` e a execução do contêiner em uma máquina local. Essa imagem de contêiner foi testada para trabalhos do Spark do AWS Glue versão 3.3. Esta imagem contém o seguinte:

- Amazon Linux
- Biblioteca ETL do AWS Glue (aws-glue-libs)
- Apache Spark 3.3.0
- Servidor de histórico do Spark
- Jupyter Lab
- Livy
- Outras dependências de bibliotecas (o mesmo conjunto que as do sistema de trabalhos do AWS Glue)

Conclua uma das seguinte seções de acordo com as suas necessidades:

- Configurar o contêiner para utilizar spark-submit
- Configurar o contêiner para utilizar o shell REPL (PySpark)
- Configurar o contêiner para utilizar o Pytest
- Configurar o contêiner para utilizar o Jupyter Lab
- Configurar o contêiner para utilizar o Visual Studio Code

Pré-requisitos

Antes de começar, verifique se o Docker está instalado e se o daemon do Docker está em execução. Para obter instruções de instalação, consulte a documentação do Docker para Mac ou Linux. A máquina que executa o Docker hospeda o contêiner do AWS Glue. Verifique também se você tem pelo menos 7 GB de espaço em disco para a imagem no host que executa o Docker.

Para obter mais informações sobre restrições ao desenvolver código do AWS Glue localmente, consulte Restrições de desenvolvimento local.

Como configurar o AWS

Para habilitar chamadas de API da AWS provenientes do contêiner, configure as credenciais da AWS seguindo as etapas. Nas seções a seguir, usaremos este perfil nomeado da AWS.

1. Configure a AWS CLI configurando um perfil nomeado. Para obter mais informações sobre a configuração da AWS CLI, consulte Configurações de arquivo de credenciais e configuração na documentação do AWS CLI.
2. Execute o seguinte comando em um terminal:

```
PROFILE_NAME="<your_profile_name>"
```

Talvez você também precise definir a variável de ambiente AWS_REGION para especificar a Região da AWS para a qual enviar as solicitações.

Configurar e executar o contêiner

O processo de configurar o contêiner para executar o código PySpark por meio do comando spark-submit inclui as seguintes etapas genéricas:

1. Extraia a imagem do Docker Hub.
2. Execute o contêiner.

Extrair a imagem do Docker Hub

Execute o seguinte comando para extrair a imagem Docker Hub:

```
docker pull amazon/aws-glue-libs:glue_libs_4.0.0_image_01
```

Executar do contêiner

Agora, você pode executar um contêiner utilizando essa imagem. É possível escolher qualquer um dos seguintes dependendo das suas necessidades.

spark-submit

Você pode executar um script de trabalho do AWS Glue executando o comando `spark-submit` no contêiner.

1. Escreva o script e salve-o como `sample1.py` no diretório `/local_path_to_workspace`. O código de exemplo está incluso como apêndice neste tópico.

```
$ WORKSPACE_LOCATION=/local_path_to_workspace
$ SCRIPT_FILE_NAME=sample.py
$ mkdir -p ${WORKSPACE_LOCATION}/src
$ vim ${WORKSPACE_LOCATION}/src/${SCRIPT_FILE_NAME}
```

2. Execute o comando a seguir para executar o comando `spark-submit` no contêiner a fim de enviar uma nova aplicação 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 corresponder ao seu ambiente. Por exemplo, você pode passar suas dependências com a configuração `--jars`. Para obter informações adicionais, consulte Iniciar aplicações com spark-submit na documentação do Spark.

Shell REPL (Pyspark)

Você pode executar o shell REPL (loops read-eval-print) para desenvolvimento interativo.

Execute o comando a seguir para executar o comando PySpark no contêiner e iniciar o shell 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 testes de unidade, você pode usar o pytest para scripts de trabalho Spark do AWS Glue.

Execute os comandos a seguir para preparação.

```
$ 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}
```

Execute o comando a seguir para executar o `pytest` no pacote de testes:

```
$ 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 ========================================================
```

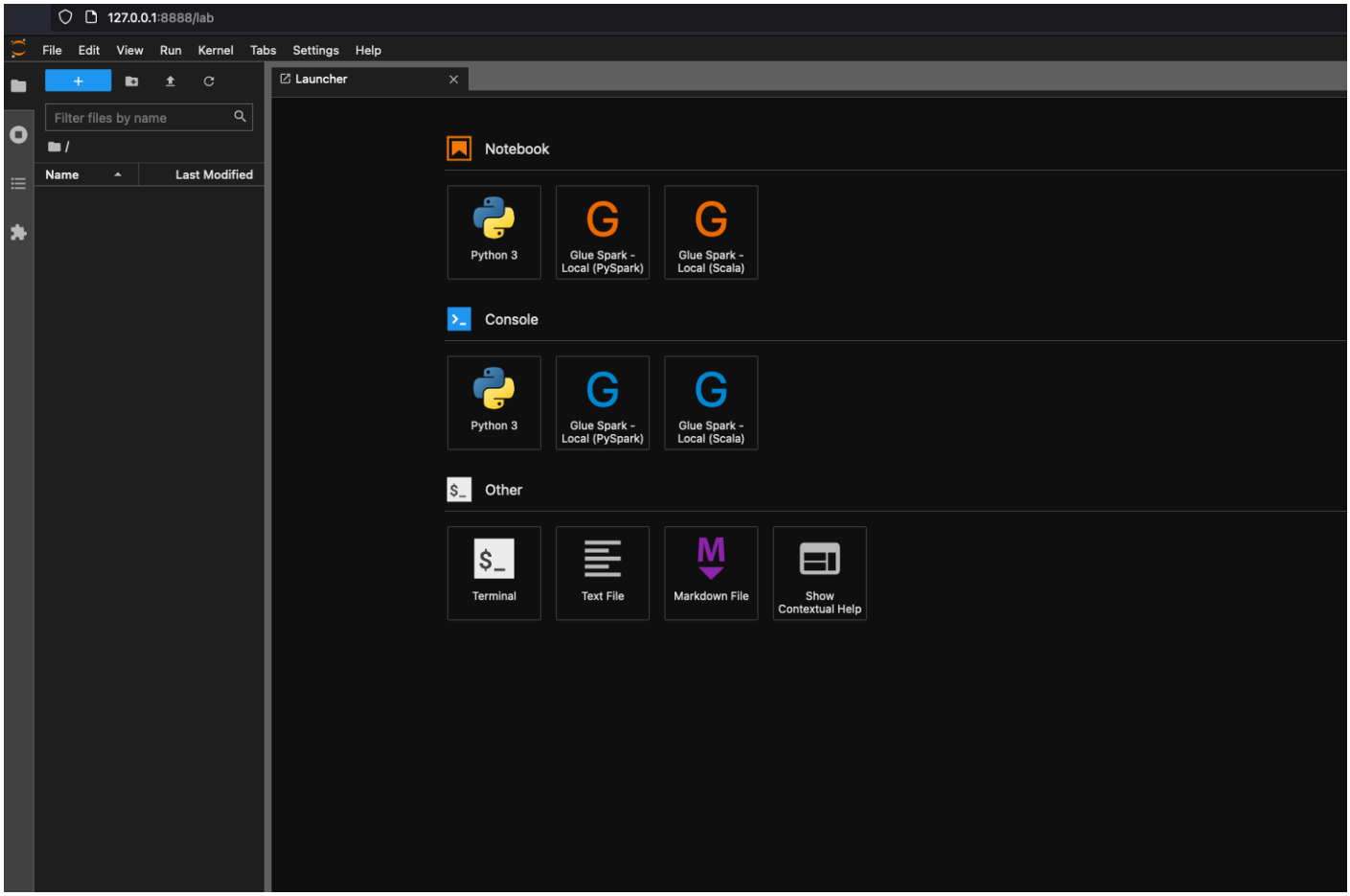
Jupyter Lab

Você pode iniciar o Jupyter para desenvolvimento interativo e consultas ad hoc em cadernos.

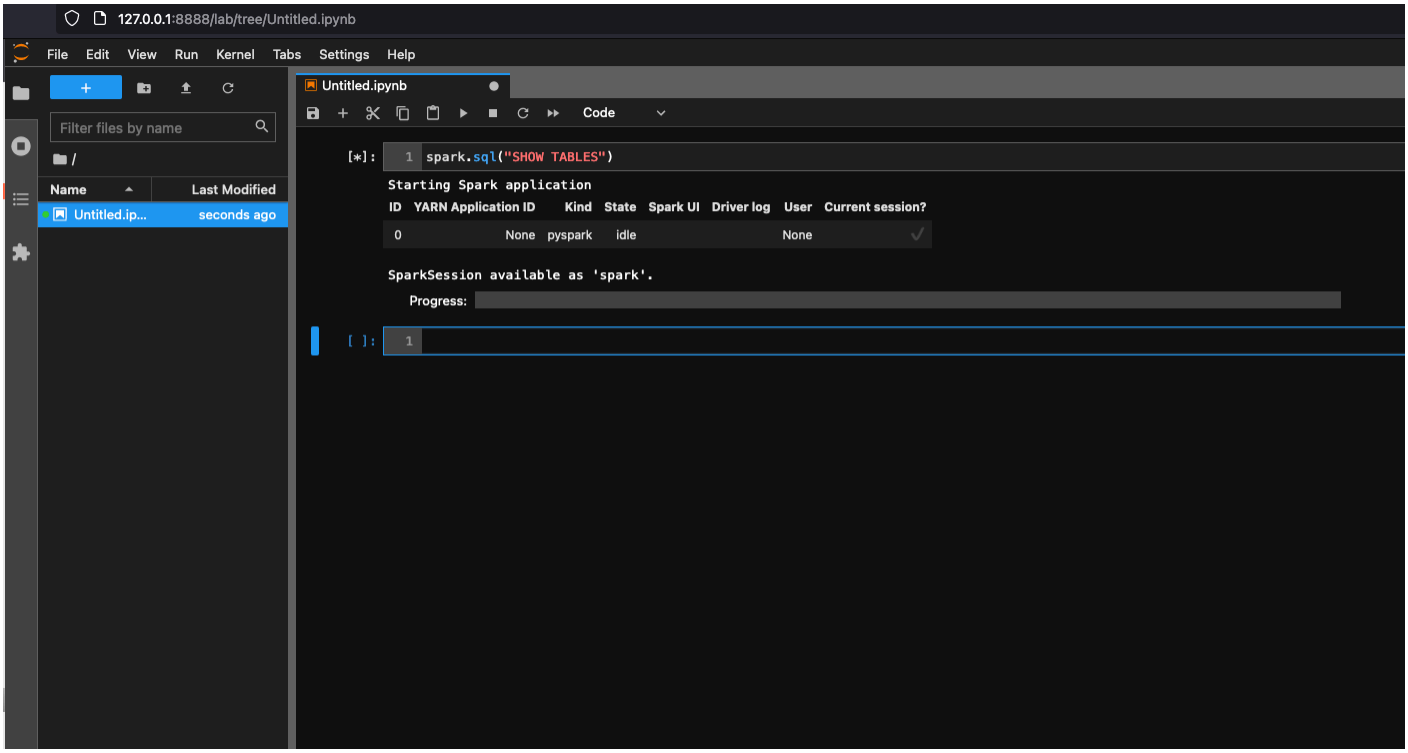
1. Execute o seguinte comando para iniciar o Jupyter Lab:

```
$ 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 no navegador da Web da máquina local para ver a interface de usuário do Jupyter lab.



3. Escolha Glue Spark Local (PySpark) em Notebook (Caderno). Você pode começar a desenvolver código na interface de usuário interativa do bloco de anotação Jupyter.



Configuração do contêiner para utilizar o Visual Studio Code

Pré-requisitos:

1. Instale o Visual Studio Code.
2. Instalar o Python.
3. Instale Visual Studio Code Remote - Containers
4. Abra a pasta de espaço de trabalho no Visual Studio Code.
5. Escolha Configurações.
6. Escolha Workspace (Espaço de trabalho).
7. Escolha Open Seettings (JSON) (Abrir configurações - JSON).
8. Cole o seguinte JSON e salve-o.

```
{
    "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/",
    ]
```

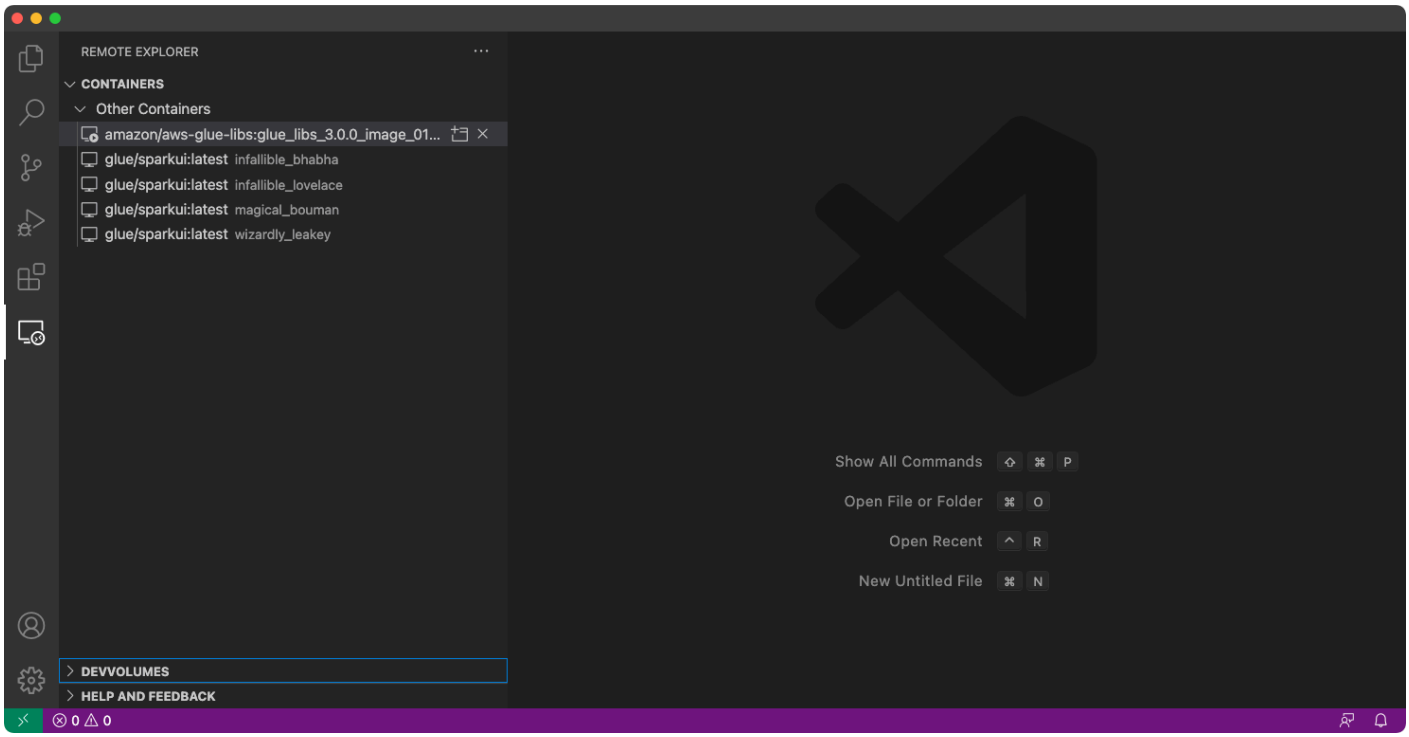
```
}
```

Etapas:

1. Execute o contêiner do Docker.

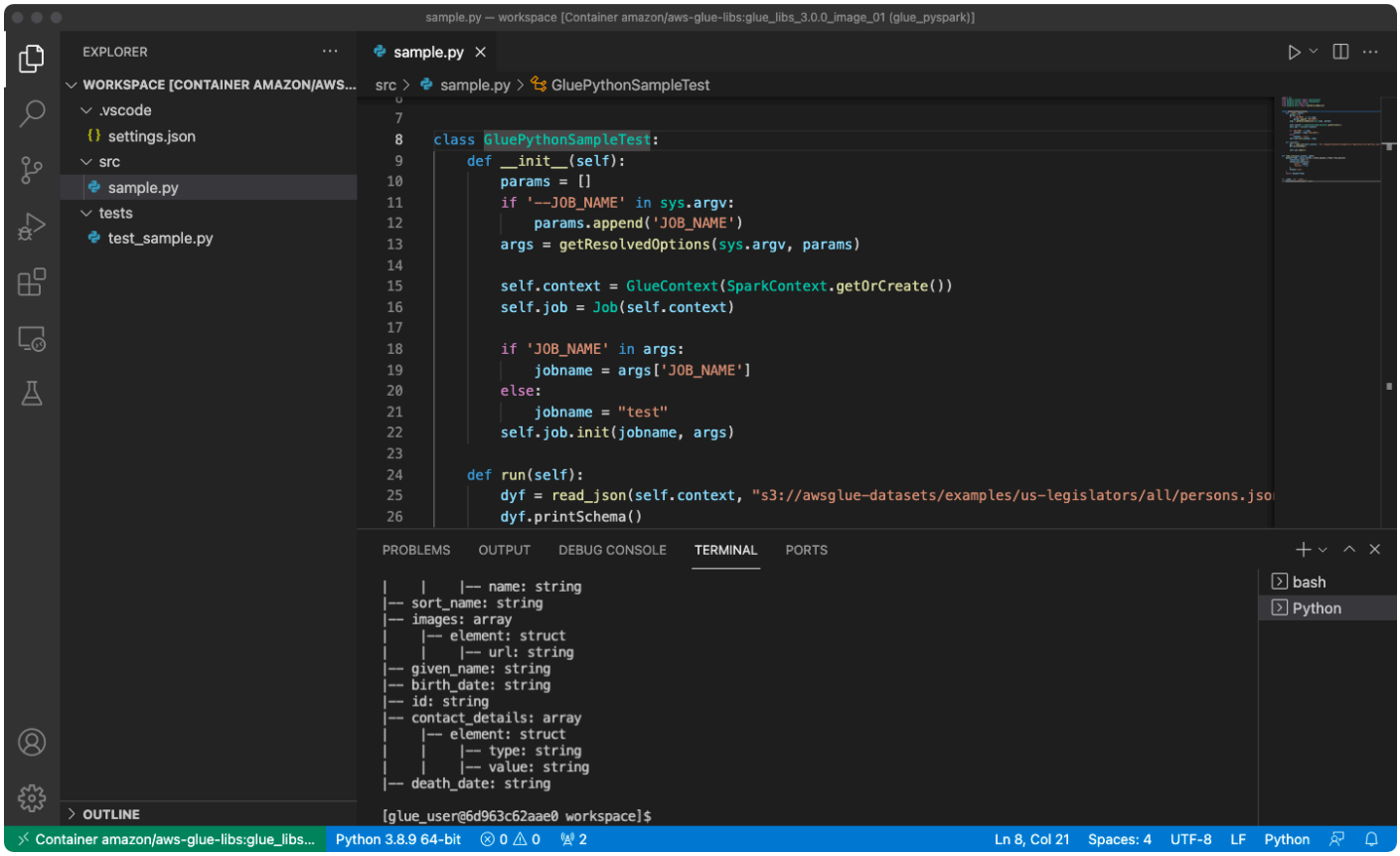
```
$ docker run -it -v ~/.aws:/home/glue_user/.aws -v $WORKSPACE_LOCATION:/
home/glue_user/workspace/ -e AWS_PROFILE=$PROFILE_NAME -e DISABLE_SSL=true
 --rm -p 4040:4040 -p 18080:18080 --name glue_pyspark amazon/aws-glue-
libs:glue_libs_4.0.0_image_01 pyspark
```

2. Inicie o Visual Studio Code.
3. Escolha Remote Explorer (Explorador remoto) no menu esquerdo e escolha `amazon/aws-glue-libs:glue_libs_4.0.0_image_01`.



4. Clique com o botão direito do mouse e escolha Attach to Container (Anexar ao contêiner). Se uma caixa de diálogo aparecer, escolha Got it (Entendido).
5. Abra o `/home/glue_user/workspace/`.
6. Crie um script PySpark do Glue e escolha Run (Executar).

   Você verá a execução do script com sucesso.



Apêndice: código de exemplo de trabalho do AWS Glue para teste

Este apêndice fornece scripts como o código de exemplo de trabalho do AWS Glue para fins de teste.

sample.py: código de exemplo para usar a biblioteca ETL do AWS Glue com uma chamada de API do 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()
```

O código acima exige permissões do Amazon S3 no AWS IAM. Você precisa conceder a política gerenciada do IAM `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess` ou uma política personalizada do IAM que permita chamar `ListBucket` e `GetObject` para o caminho do Amazon S3.

test_sample.py: código de exemplo para o teste de unidade 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
```

Desenvolvimento com a biblioteca ETL do AWS Glue

A biblioteca ETL do AWS Glue está disponível em um bucket público do Amazon S3 e pode ser consumida pelo sistema de compilação Apache Maven. Isso permite que você desenvolva e teste seus scripts de extração, transformação e carregamento (ETL) Python e Scala localmente, sem a necessidade de uma conexão de rede. O desenvolvimento local com a imagem do Docker é recomendado, pois fornece um ambiente configurado adequadamente para o uso dessa biblioteca.

Desenvolvimento local está disponível para todas as versões do AWS Glue, incluindo o AWS Glue versão 0.9, 1.0, 2.0 e posteriores. Para obter informações sobre as versões do Python e do Apache Spark que estão disponíveis com o AWS Glue, consulte o Glue version job property.

A biblioteca é liberada com a licença de software da Amazon (https://aws.amazon.com/asl).

Restrições de desenvolvimento local

Lembre-se das seguintes restrições ao usar a biblioteca Scala do AWS Glue para desenvolvimento local.

- Evite criar um conjunto jar ("fat jar" ou "uber jar") com a biblioteca do AWS Glue, pois isso fará com que os seguintes recursos sejam desabilitados:
  - Marcadores de trabalho
  - Gravador AWS Glue Parquet (Uso do formato Parquet no AWS Glue)
  - Transformação FillMissingValues (Scala ou Python)

  Esses recursos estão disponíveis somente no sistema de trabalhos AWS Glue.
- Não há suporte para a transformação FindMatches com desenvolvimento local.
- O leitor de CSV SIMD vetorizado não é compatível com desenvolvimento local.
- A propriedade customJdbcDrivers3Path para carregar o driver JDBC do caminho do S3 não é compatível com desenvolvimento local. Como alternativa, você pode baixar o driver JDBC em seu local e carregá-lo daí.
- O Glue Data Quality não é compatível com desenvolvimento local.

Desenvolver localmente com o Python

Conclua algumas etapas de pré-requisito e use os utilitários do AWS Glue para testar e enviar seu script de ETL Python.

Pré-requisitos para o desenvolvimento local com o Python

Execute estas etapas para se preparar para o desenvolvimento local do Python:

1. Clone o repositório de Python do AWS Glue do GitHub (https://github.com/awslabs/aws-glue-libs).
2. Execute um destes procedimentos:
   - Para o AWS Glue versão 0.9, confira o branch `glue-0.9`.
   - Para o AWS Glue versão 1.0, confira o branch `glue-1.0`. Todas as versões posteriores ao AWS Glue 0.9 oferecem suporte a Python 3.
   - Para o AWS Glue versão 2.0, confira o branch `glue-2.0`.
   - Para o AWS Glue versões 3.0, confira a ramificação `glue-3.0`.
   - Para o AWS Glue versão 4.0, confira a ramificação `master`.

3. Instale o Apache Maven do seguinte local: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-common/apache-maven-3.6.0-bin.tar.gz.
4. Instale a distribuição do Apache Spark de um dos seguintes locais:
   - Para o AWS Glue versão 0.9: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-0.9/spark-2.2.1-bin-hadoop2.7.tgz
   - Para o AWS Glue versão 1.0: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-1.0/spark-2.4.3-bin-hadoop2.8.tgz
   - Para o AWS Glue versão 2.0: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-2.0/spark-2.4.3-bin-hadoop2.8.tgz
   - Para o AWS Glue versão 3.0: 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 o AWS Glue versão 4.0: 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 a variável de ambiente `SPARK_HOME`, definindo-a como o local raiz extraído do arquivo do Spark. Por exemplo:
   - Para o AWS Glue versão 0.9: `export SPARK_HOME=/home/$USER/spark-2.2.1-bin-hadoop2.7`
   - Para o AWS Glue versão 1.0 e 2.0: `export SPARK_HOME=/home/$USER/spark-2.4.3-bin-spark-2.4.3-bin-hadoop2.8`
   - Para o AWS Glue versão 3.0: `export SPARK_HOME=/home/$USER/spark-3.1.1-amzn-0-bin-3.2.1-amzn-3`
   - Para o AWS Glue versão 4.0: `export SPARK_HOME=/home/$USER/spark-3.3.0-amzn-1-bin-3.3.3-amzn-0`

Executar o script Python de ETL

Com os arquivos jar do AWS Glue disponíveis para desenvolvimento local, é possível executar o pacote do AWS Glue Python localmente.

Use os seguintes utilitários e estruturas para testar e executar seu script do Python. Os comandos listados na tabela a seguir são executados no diretório raiz do pacote Python do AWS Glue.

Utilitário	Command	Descrição
Shell do AWS Glue	`./bin/gluepyspark`	Insira e execute scripts Python em um shell que se integre com bibliotecas de ETL do AWS Glue.
Submit (Enviar) AWS Glue.	`./bin/gluesparksubmit`	Envie um script completo do Python para execução.
Pytest	`./bin/gluepytest`	Escreva e execute testes de unidade do seu código do Python. O módulo pytest deve estar instalado e disponível no `PATH`. Para obter mais informações, consulte a documentação do pytest.

Desenvolver localmente com o Scala

Conclua algumas etapas de pré-requisito e emita um comando Maven para executar seu script de ETL Scala localmente.

Pré-requisitos para o desenvolvimento local com o Scala

Execute estas etapas para se preparar para o desenvolvimento local com o Scala.

Etapa 1: instalar o software

Nesta etapa, instale o software e defina a variável de ambiente necessária.

1. Instale o Apache Maven do seguinte local: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-common/apache-maven-3.6.0-bin.tar.gz.
2. Instale a distribuição do Apache Spark de um dos seguintes locais:
   - Para o AWS Glue versão 0.9: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-0.9/spark-2.2.1-bin-hadoop2.7.tgz
   - Para o AWS Glue versão 1.0: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-1.0/spark-2.4.3-bin-hadoop2.8.tgz
   - Para o AWS Glue versão 2.0: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-2.0/spark-2.4.3-bin-hadoop2.8.tgz

- Para o AWS Glue versão 3.0: 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 o AWS Glue versão 4.0: 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 a variável de ambiente `SPARK_HOME`, definindo-a como o local raiz extraído do arquivo do Spark. Por exemplo:
   - Para o AWS Glue versão 0.9: `export SPARK_HOME=/home/$USER/spark-2.2.1-bin-hadoop2.7`
   - Para o AWS Glue versão 1.0 e 2.0: `export SPARK_HOME=/home/$USER/spark-2.4.3-bin-spark-2.4.3-bin-hadoop2.8`
   - Para o AWS Glue versão 3.0: `export SPARK_HOME=/home/$USER/spark-3.1.1-amzn-0-bin-3.2.1-amzn-3`
   - Para o AWS Glue versão 4.0: `export SPARK_HOME=/home/$USER/spark-3.3.0-amzn-1-bin-3.3.3-amzn-0`

Etapa 2: configurar o projeto Maven

Use o arquivo `pom.xml` a seguir como um modelo para seus aplicativos do AWS Glue Scala. Ele contém os elementos `dependencies`, `repositories` e `plugins` necessários. Substitua a string `Glue version` por um dos seguintes itens:

- `4.0.0` para o AWS Glue versão 4.0
- `3.0.0` para o AWS Glue versão 3.0
- `1.0.0` para o AWS Glue versão 1.0 e 2.0
- `0.9.0` para o AWS Glue versão 0.9

```
  <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>
```

Executar o script Scala de ETL

Execute o seguinte comando no diretório raiz do projeto Maven para executar seu script de ETL em Scala.

```
mvn exec:java -Dexec.mainClass="mainClass" -Dexec.args="--JOB-NAME jobName"
```

Substitua *mainClass* pelo nome totalmente qualificado da classe principal do script. Substitua *jobName* pelo nome do trabalho desejado.

Configurar um ambiente de teste

Para ver exemplos de configuração de um ambiente de teste local, consulte os seguintes artigos do blog:

- Desenvolvimento de um pipeline de ETL do AWS Glue localmente sem uma conta da AWS
- Desenvolvimento de trabalhos de ETL do AWS Glue localmente usando um contêiner

Se você quiser usar endpoints de desenvolvimento ou cadernos para testar seus scripts de ETL, consulte Desenvolver scripts com endpoints de desenvolvimento.

> **Note**
>
> Os endpoints de desenvolvimento não são suportados para uso com trabalhos do AWS Glue versão 2.0. Para obter mais informações, consulte Executar trabalhos de ETL do Spark com startup reduzidos.

Endpoints de desenvolvimento

> **Note**
>
> A experiência do console para endpoints de desenvolvimento foi removida em 31 de março de 2023. A criação, a atualização e o monitoramento de endpoints de desenvolvimento ainda estão disponíveis por meio do API de endpoints de desenvolvimento e da CLI do AWS Glue.

É altamente recomendável migrar de endpoints de desenvolvimento para sessões interativas pelos motivos listados abaixo. Para ver as ações necessárias para migrar endpoints de desenvolvimento para sessões interativas, consulte Migrar de endpoints de desenvolvimento para sessões interativas.

Descrição	Endpoints de desenvolvimento	Sessões interativas
Compatibilidade com versão do Glue	Compatível com o AWS Glue versões 0.9 e 1.0	Compatível com o AWS Glue versões 2.0 e posteriores
A partir de agora, os endpoints de desenvolvimento não	No momento, sessões interativas não estão	

Descrição	Endpoints de desenvolvimento	Sessões interativas
estarão disponíveis nas regiões Ásia-Pacífico (Jacarta) (`ap-southeast-3`), Oriente Médio (EAU) (`me-central-1`) Europa (Espanha) (`eu-south-2`), Europa (Zurique) (`eu-central-2`) nem outras regiões novas	disponíveis na região Oriente Médio (EAU) (`me-central-1`), mas podem ser disponibilizadas posteriormente	
Método de acesso ao cluster do Spark	Compatível com SSH, shell do REPL, caderno Jupyter, IDE (por exemplo, PyCharm)	Compatível com caderno AWS Glue Studio, caderno Jupyter, vários IDEs (por exemplo, Visual Studio Code, PyCharm) e caderno SageMaker
Tempo até o primeiro byte	Requer de 10 a 15 minutos para configurar um cluster do Spark	Pode levar até 1 minuto para configurar um cluster efêmero do Spark

Descrição	Endpoints de desenvolvimento	Sessões interativas
O modelo de preços	A AWS cobra pelos endpoints de desenvolvimento com base no tempo que o endpoint é provisionado e no número de DPUs. Os endpoints de desenvolvimento não têm tempo limite. Há uma duração mínima para cobrança de 10 minutos para cada endpoint de desenvolvimento provision ado. Além disso, a AWS cobra pelos instâncias do caderno Jupyter no Amazon EC2 e pelos caderno do SageMaker quando você os configura com o endpoint de desenvolv imento.	A AWS cobra por sessões interativas com base no tempo que a sessão permanece ativa e no número de DPUs. As sessões interativas têm tempos limites de inatividade configuráveis. Os cadernos do AWS Glue Studio oferecem uma interface integrada para sessões interativas e são oferecidos sem custo adicional . Há uma duração mínima para cobrança de 1 minuto para cada sessão interativa. Os cadernos do AWS Glue Studio fornecem uma interface embutida para sessões interativas e são oferecidos sem custo adicional
Experiência de console	Disponível somente por meio da CLI e da API	Disponível por meio do console, da CLI e de APIs do AWS Glue

Migrar de endpoints de desenvolvimento para sessões interativas

Use a lista de verificação a seguir para determinar o método apropriado para migrar de endpoints de desenvolvimento para sessões interativas.

Seu script depende de recursos específicos do AWS Glue 0.9 ou 1.0 (por exemplo, HDFS, YARN etc.)?

Se a resposta for sim, consulte Migrar trabalhos do AWS Glue para o AWS Glue versão 3.0 para saber como migrar do Glue 0.9 ou 1.0 para o Glue 3.0 e versões posteriores.

Qual método você usa para acessar o endpoint de desenvolvimento?

Se você usar este método	Então faça isso
Caderno do SageMaker, caderno Jupyter ou JupyterLab	Migre para o caderno AWS Glue Studio baixando os arquivos `.ipynb` no Jupyter e crie um novo trabalho de caderno AWS Glue Studio carregando o arquivo `.ipynb`. Ou então, você pode usar o SageMaker Studio e selecionar o kernel do AWS Glue.
Notebook do Zeppelin	Converta o caderno em um caderno Jupyter manualmente, copiando e colando o código, ou automaticamente, usando um conversor de terceiros, como o ze2nb. Depois, use o caderno no caderno do AWS Glue Studio ou do SageMaker Studio.
IDE	Consulte Criar trabalhos do AWS Glue com o PyCharm usando as sessões interativas do AWS Glue ou Usar sessões interativas com o código do Microsoft Visual Studio.
RESPOSTA	Instale o `aws-glue-session package` localmente e depois execute o seguinte comando: • Para Python: `jupyter console --kernal glue_pyspark` • Para Scala: `jupyter console --kernal glue_spark`
SSH	Não há nenhuma opção correspondente nas sessões interativas. Como alternativa, você pode usar uma imagem do Docker. Para saber mais, consulte Desenvolver usando com uma imagem do Docker.

As seções a seguir fornecem mais informações sobre o uso de endpoints de desenvolvimento para desenvolver trabalhos no AWS Glue versão 1.0.

Tópicos

Desenvolver scripts com endpoints de desenvolvimento

> **Note**
>
> Os endpoints de desenvolvimento só são compatíveis com as versões do AWS Glue anteriores à 2.0. Para um ambiente interativo onde você possa criar e testar scripts de ETL, use Notebooks on AWS Glue Studio.

O AWS Glue pode criar um ambiente, conhecido como endpoint de desenvolvimento, que você pode usar para desenvolver e testar iterativamente seus scripts de extração, transformação e carregamento (ETL). Você pode criar, editar e excluir endpoints de desenvolvimento usando o console ou a API do AWS Glue.

Gerenciar o ambiente de desenvolvimento

Ao criar um endpoint de desenvolvimento, você fornece valores de configuração para provisionar o ambiente de desenvolvimento. Esses valores informam ao AWS Glue como configurar a rede para que você possa acessar o endpoint de forma segura e o endpoint possa acessar seus armazenamentos de dados.

Depois, é possível criar um notebook que se conecte ao endpoint e usá-lo para criar e testar seu script de ETL. Quando estiver satisfeito com os resultados do seu processo de desenvolvimento, você poderá criar um trabalho de ETL que execute seu script. Com esse processo, você pode adicionar funções e depurar seus scripts de forma interativa.

Siga os tutoriais nesta seção para saber como usar seu endpoint de desenvolvimento com notebooks.

Tópicos

- Como os endpoints de desenvolvimento do AWS Glue funcionam com cadernos do SageMaker
- Adicionar um endpoint de desenvolvimento
- Acessar o endpoint de desenvolvimento
- Tutorial: configurar um caderno Jupyter no JupyterLab para testar e depurar scripts de ETL
- Tutorial: usar um caderno do SageMaker com seu endpoint de desenvolvimento
- Tutorial: usar um shell REPL com seu endpoint de desenvolvimento
- Tutorial: configurar um PyCharm Professional com um endpoint de desenvolvimento
- Configuração avançada: compartilhar endpoints de desenvolvimento entre vários usuários

Fluxo de trabalho do endpoint de desenvolvimento

Para usar um endpoint de desenvolvimento do AWS Glue, você pode seguir este fluxo de trabalho:

1. Crie um endpoint de desenvolvimento usando o console ou a API. O endpoint é executado em uma nuvem privada virtual (VPC) com seus grupos de segurança definidos.
2. O console ou a API sonda o endpoint de desenvolvimento até que ele seja provisionado e esteja pronto para operar. Quando estiver pronto, conecte-se ao endpoint de desenvolvimento usando um dos seguintes métodos para criar e testar scripts do AWS Glue.
   - Crie um caderno do SageMaker em sua conta. Para obter mais informações sobre como criar um notebook, consulte the section called “Criação de código com notebooks AWS Glue Studio”.
   - Abra uma janela de terminal para se conectar diretamente a um endpoint de desenvolvimento.
   - Se tiver a versão profissional do PyCharm Python IDE do JetBrains, conecte-a a um endpoint de desenvolvimento e use-a para desenvolver de forma interativa. Se você inserir as instruções `pydevd` no seu script, o PyCharm poderá oferecer suporte a pontos de interrupção remotos.
3. Ao concluir a depuração e o teste no seu endpoint de desenvolvimento, você poderá excluí-lo.

Como os endpoints de desenvolvimento do AWS Glue funcionam com cadernos do SageMaker

Uma das maneiras comuns de acessar seus endpoints de desenvolvimento é usar o Jupyter em cadernos do SageMaker. O caderno Jupyter é um aplicação Web de código aberto que é

amplamente utilizada em visualização, análise, machine learning etc. Um caderno do SageMaker do AWS Glue oferece uma experiência de caderno Jupyter com endpoints de desenvolvimento do AWS Glue. No caderno do SageMaker do AWS Glue, o ambiente de caderno Jupyter é pré-configurado com SparkMagic, um plugin de código aberto do Jupyter para enviar trabalhos do Spark para um cluster remoto do Spark. Apache Livy é um serviço que permite a interação com um cluster remoto do Spark por meio de uma API REST. No caderno do SageMaker do AWS Glue, o SparkMagic está configurado para chamar a API REST em um servidor Livy em execução em um endpoint de desenvolvimento do AWS Glue.

O fluxo de texto a seguir explica como cada componente funciona:

Notebook do SageMaker do AWS Glue: (Jupyter → SparkMagic) → (rede) → endpoint de desenvolvimento do AWS Glue: (Apache Livy → Apache Spark)

Depois de executar o script do Spark escrito em cada parágrafo em um caderno Jupyter, o código do Spark é enviado ao servidor Livy via SparkMagic e, em seguida, um trabalho do Spark chamado “livy-session-N” é executado no cluster do Spark. Esse trabalho é chamado de sessão do Livy. O trabalho do Spark será executado enquanto a sessão do caderno estiver ativa. O trabalho do Spark será terminado quando você desligar o kernel do Jupyter do caderno, ou quando o tempo da sessão se esgotar. Um trabalho do Spark é iniciado por arquivo de caderno (.ipynb).

Você pode usar um único endpoint de desenvolvimento do AWS Glue com várias instâncias de cadernos do SageMaker. Você pode criar vários arquivos de caderno em cada instância de cadernos do SageMaker. Quando você abre um arquivo de cada caderno e executa os parágrafos, uma sessão do Livy é iniciada por arquivo de caderno no cluster do Spark via SparkMagic. Cada sessão do Livy corresponde a um único trabalho do Spark.

Comportamento padrão para endpoints de desenvolvimento do AWS Glue e cadernos do SageMaker

Os trabalhos do Spark são executados com base na configuração do Spark. Existem várias maneiras de definir a configuração do Spark (por exemplo, configuração do cluster do Spark, configuração do SparkMagic etc.).

Por padrão, o Spark aloca recursos de cluster para uma sessão do Livy com base na configuração do cluster do Spark. Em endpoints de desenvolvimento do AWS Glue, a configuração do cluster depende do tipo de operador. Esta tabela explica as configurações comuns por tipo de operador.

	Padrão	G.1X	G.2X
`spark.driver.memory`	5G	10G	20G
`spark.executor.memory`	5G	10G	20G
`spark.executor.cores`	4	8	16
`spark.dynamicAllocation.enabled`	VERDADEIRO	VERDADEIRO	VERDADEIRO

O número máximo de executores do Spark é calculado automaticamente pela combinação de DPU (ou `NumberOfWorkers`) e tipo de operador.

	Padrão	G.1X	G.2X
O número máximo de executores do Spark	`(DPU - 1) * 2 - 1`	`(NumberOfWorkers - 1)`	`(NumberOfWorkers - 1)`

Por exemplo, se o endpoint de desenvolvimento tiver dez operadores e o tipo de operador for `G.1X`, então você terá nove executores do Spark e todo o cluster terá 90G de memória de executor, uma vez que cada executor terá 10G de memória.

Independentemente do tipo de operador especificado, a alocação dinâmica de recursos do Spark será ativada. Se um conjunto de dados for grande o suficiente, o Spark pode alocar todos os executores em uma única sessão do Livy, já que `spark.dynamicAllocation.maxExecutors`

não é definido por padrão. Isso significa que outras sessões do Livy no mesmo endpoint de desenvolvimento esperarão para iniciar novos executores. Se o conjunto de dados for pequeno, o Spark poderá alocar executores em várias sessões do Livy ao mesmo tempo.

> **Note**
>
> Para obter mais informações sobre como os recursos são alocados em diferentes casos de uso e como você define uma configuração para modificar o comportamento, consulte Configuração avançada: compartilhar endpoints de desenvolvimento entre vários usuários.

Adicionar um endpoint de desenvolvimento

Use endpoints de desenvolvimento para desenvolver e testar iterativamente seus scripts de extração, transformação e carregamento (ETL) no AWS Glue. Só é possível trabalhar com endpoints de desenvolvimento por meio do AWS Command Line Interface.

1. Em uma janela de linha de comando, insira um comando semelhante ao seguinte.

   ```
   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"
   ```

   Esse comando especifica o AWS Glue versão 1.0. Como essa versão oferece suporte ao Python 2 e ao Python 3, você pode usar o parâmetro `arguments` para indicar a versão desejada do Python. Se o parâmetro `glue-version` for omitido, o AWS Glue versão 0.9 será assumido. Para obter mais informações sobre as versões do AWS Glue, consulte o Glue version job property.

   Para obter informações sobre parâmetros de linha de comando adicionais, consulte create-dev-endpoint na Referência de comandos da AWS CLI.

2. (Opcional) Insira o comando a seguir para verificar o status do endpoint de desenvolvimento. Quando o status mudar para `READY`, o endpoint de desenvolvimento estará pronto para uso.

   ```
   aws glue get-dev-endpoint --endpoint-name "endpoint1"
   ```

Acessar o endpoint de desenvolvimento

Quando você cria um endpoint de desenvolvimento em uma nuvem privada virtual (VPC), o AWS Glue retorna apenas um endereço IP privado. O campo de endereço IP público não é preenchido. Quando você cria um endpoint de desenvolvimento não seja da VPC, o AWS Glue retorna somente um endereço IP público.

Se o endpoint de desenvolvimento tiver um Public address (Endereço público), confirme se ele está acessível com a chave privada SSH do endpoint de desenvolvimento, como no exemplo a seguir.

```
ssh -i dev-endpoint-private-key.pem glue@public-address
```

Suponha que o endpoint de desenvolvimento tenha um Private address (Endereço privado), a sub-rede da VPC seja roteável pela Internet pública e seus grupos de segurança permitam acesso de entrada do cliente. Nesse caso, siga estas etapas para anexar um Elastic IP address (Endereço IP elástico) a um endpoint de desenvolvimento para permitir o acesso na Internet.

> **Note**
>
> Se você deseja usar endereços IP elásticos, a sub-rede que está sendo usada requer um gateway de Internet associado por meio da tabela de rotas.

Como acessar um endpoint de desenvolvimento anexando um endereço IP elástico

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Dev endpoints (Endpoints de desenvolvimento) e navegue até a página de detalhes do endpoint de desenvolvimento. Registre o Private address (Endereço privado) a ser usado na próxima etapa.
3. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
4. No painel de navegação, em Network & Security (Rede e segurança), selecione Network Interfaces (Interfaces de rede).
5. Pesquise o Private DNS (IPv4) (DNS privado (IPv4)) que corresponde ao Private address (Endereço privado) na página de detalhes do endpoint de desenvolvimento do console do AWS Glue.

Pode ser necessário alterar as colunas que são exibidas no console do Amazon EC2. Anote o Network interface ID (ID da interface de rede) (ENI) desse endereço (por exemplo, `eni-12345678`).

6. No console do Amazon EC2, em Network & Security (Rede e segurança), escolha Elastic IPs (IPs elásticos).
7. Escolha Allocate new address (Alocar novo endereço) e, depois, escolha Allocate (Alocar) para alocar um novo endereço IP elástico.
8. Na página Elastic IPs (IPs elásticos), selecione o Elastic IP (IP elástico) recém-alocado. Em seguida, selecione Actions (Ações) e Associate address (Associar endereço).
9. Na página Associate address (Associar endereço), faça o seguinte:
   - Em Resource type (Tipo de recurso), selecione Network interface (Interface de rede).
   - No campo Network interface (Interface de rede), digite o Network interface ID (ID da interface de rede) (ENI) do endereço privado.
   - Selecione Associar.
10. Confirme se o endereço IP elástico recém-associado está acessível com a chave privada SSH associada ao endpoint de desenvolvimento, como no exemplo a seguir.

```
ssh -i dev-endpoint-private-key.pem glue@elastic-ip
```

Para obter informações sobre como usar um bastion host para obter acesso SSH ao endereço privado do endpoint de desenvolvimento, consulte o post do blog de segurança da AWS Securely Connect to Linux Instances Running in a Private Amazon VPC.

Tutorial: configurar um caderno Jupyter no JupyterLab para testar e depurar scripts de ETL

Neste tutorial, você conectará um caderno Jupyter no JupyterLab em execução em sua máquina local a um endpoint de desenvolvimento. Você o faz para que possa executar, depurar e testar interativamente scripts de extração, transformação e carregamento (ETL) do AWS Glue antes de implantá-los. Este tutorial usa o redirecionamento de portas Secure Shell (SSH) para conectar sua máquina local a um endpoint de desenvolvimento do AWS Glue. Para obter mais informações, consulte Redirecionamento de portas na Wikipédia.

Etapa 1: Instalar o JupyterLab e o Sparkmagic

Você pode instalar o JupyterLab usando o `conda` ou `pip`. O `conda` é um sistema de gerenciamento de pacotes e de ambientes de código aberto que é executado no Windows, macOS e Linux. Já o `pip` é o instalador de pacotes para Python.

Se estiver instalando no macOS, é necessário ter o Xcode instalado antes de instalar o Sparkmagic.

1. Instale o JupyterLab, o Sparkmagic e as extensões 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. Verifique o diretório `sparkmagic` de `Location`.

```
$ pip show sparkmagic | grep Location
Location: /Users/username/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-
packages
```

3. Altere seu diretório para o mesmo retornado para `Location` e instale os kernels para Scala e 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. Faça download de um arquivo de amostra `config`.

```
$ curl -o ~/.sparkmagic/config.json https://raw.githubusercontent.com/jupyter-
incubator/sparkmagic/master/sparkmagic/example_config.json
```

   Neste arquivo de configuração, você pode configurar parâmetros relacionados ao Spark, como `driverMemory` e `executorCores`.

Etapa 2: Iniciar o JupyterLab

Quando você inicia o JupyterLab, seu navegador da Web padrão é aberto automaticamente e o URL `http://localhost:8888/lab/workspaces/{workspace_name}` é exibido.

```
$ jupyter lab
```

Etapa 3: iniciar o redirecionamento de porta SSH para se conectar ao endpoint de desenvolvimento

Em seguida, use o encaminhamento de porta local SSH para encaminhar uma porta local (aqui, 8998) ao destino remoto definido por AWS Glue (`169.254.76.1:8998`).

1. Abra uma janela do terminal separada que conceda acesso ao SSH. No Microsoft Windows, você pode usar o shell BASH fornecido pelo Git for Windows (Git para Windows) ou instalar o Cygwin.
2. Execute o seguinte comando SSH, modificado da seguinte forma:

   - Substitua *`private-key-file-path`* pelo caminho do arquivo `.pem` que contém a chave privada correspondente à chave pública que você usou para criar seu endpoint de desenvolvimento.
   - Se você estiver redirecionando uma porta diferente de 8998, substitua 8998 pelo número da porta que está de fato usando localmente. O endereço `169.254.76.1:8998` é a porta remota e não é alterada por você.
   - Substitua *`dev-endpoint-public-dns`* pelo endereço DNS público do seu endpoint de desenvolvimento. Para encontrar esse endereço, navegue até seu endpoint de desenvolvimento no console do AWS Glue, escolha o nome e copie o valor de Public address (Endereço público) listado na página Endpoint details (Detalhes do endpoint).

   ```
   ssh -i private-key-file-path -NTL 8998:169.254.76.1:8998 glue@dev-endpoint-public-
   dns
   ```

   Você provavelmente verá uma mensagem de aviso semelhante à seguinte:

   ```
   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)?
   ```

   Digite **`yes`** (sim) e deixe aberta a janela do terminal enquanto utiliza o JupyterLab.
3. Verifique se o redirecionamento de porta SSH está funcionando corretamente com o endpoint de desenvolvimento.

```
$ curl localhost:8998/sessions
{"from":0,"total":0,"sessions":[]}
```

Etapa 4: executar um fragmento de script simples em um parágrafo do caderno

Agora seu caderno no JupyterLab deve funcionar com seu endpoint de desenvolvimento. Digite o seguinte fragmento de script no seu caderno e execute-o.

1. Verifique se o Spark está sendo executado com êxito. O comando a seguir instrui o Spark a calcular 1 e, em seguida, imprimir o valor.

   ```
   spark.sql("select 1").show()
   ```

2. Verifique se a integração com o AWS Glue Data Catalog está funcionando. O comando a seguir lista as tabelas do Data Catalog.

   ```
   spark.sql("show tables").show()
   ```

3. Verifique se um fragmento de script simples que usa as bibliotecas do AWS Glue funciona.

   O script a seguir usa os metadados da tabela `persons_json` no AWS Glue Data Catalog para criar um `DynamicFrame` dos seus dados de exemplo. Em seguida, ele imprime a contagem de itens e o esquema desses dados.

```
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()
```

O resultado do script é o seguinte:

```
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
```

Solução de problemas

- Durante a instalação do JupyterLab, se o seu computador estiver atrás de um proxy corporativo ou firewall, você poderá encontrar erros de HTTP e SSL devido a perfis de segurança personalizados gerenciados por departamentos de TI corporativos.

  Segue-se um exemplo de um erro típico que ocorre quando `conda` não consegue se conectar aos seus próprios repositórios:

```
CondaHTTPError: HTTP 000 CONNECTION FAILED for url <https://repo.anaconda.com/pkgs/
main/win-64/current_repodata.json>
```

Isso pode acontecer porque sua empresa pode bloquear conexões com repositórios amplamente utilizados em comunidades de Python e JavaScript. Para obter mais informações, consulte Installation Problems (Problemas de instalação) no site do JupyterLab.

- Se encontrar um erro de conexão recusada ao tentar se conectar ao seu endpoint de desenvolvimento, pode ser que você esteja usando um endpoint de desenvolvimento desatualizado. Tente criar um novo endpoint de desenvolvimento e se reconectar.

Tutorial: usar um caderno do SageMaker com seu endpoint de desenvolvimento

No AWS Glue, você pode criar um endpoint de desenvolvimento e, depois, criar um caderno do SageMaker para ajudar a desenvolver os scripts de ETL e machine learning. Um bloco de anotações do SageMaker é uma instância de computação de machine learning totalmente gerenciada que executa o aplicativo Jupyter Notebook.

1. No console do AWS Glue, escolha Dev endpoints para navegar até a lista de endpoints de desenvolvimento.
2. Marque a caixa de seleção ao lado do nome de um endpoint de desenvolvimento que você deseja usar e, no menu Action (Ação), escolha Create SageMaker notebook (Criar bloco de anotações do SageMaker).
3. Preencha a página Create and configure a notebook (Criar e configurar um bloco de anotações) da seguinte forma:

   a. Insira um nome de bloco de anotações.

   b. Em Attach to development endpoint (Anexar ao endpoint de desenvolvimento), verifique o endpoint de desenvolvimento.

   c. Crie ou escolha uma função do AWS Identity and Access Management (IAM).

      A criação de uma função é recomendada. Se você usar uma função existente, verifique se ela tem as permissões necessárias. Para ter mais informações, consulte the section called “Etapa 6: criar uma política do IAM para cadernos do SageMaker”.

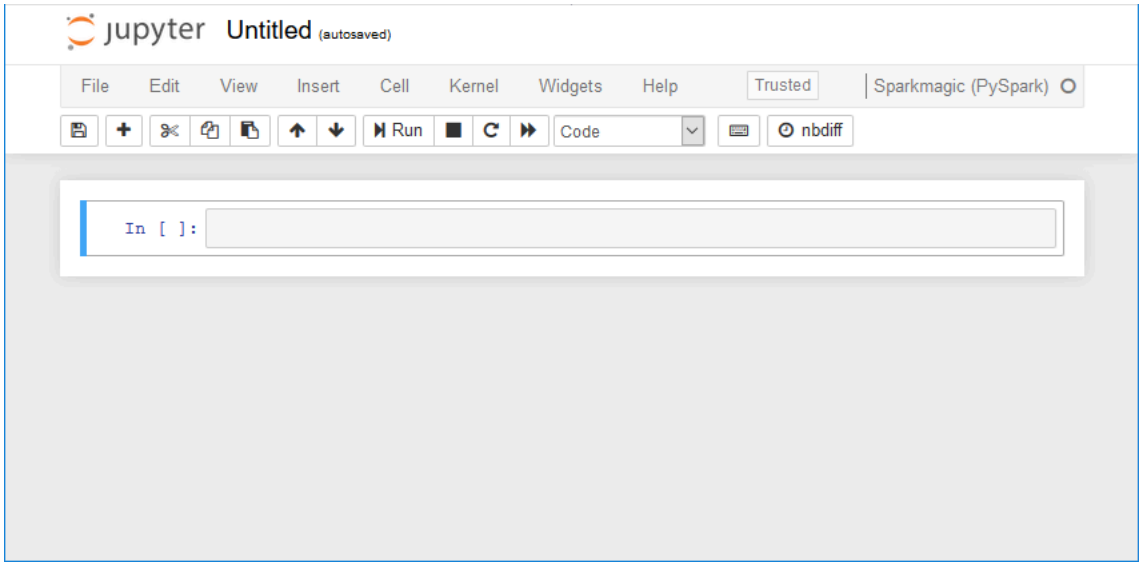
   d. (Opcional) Escolha uma VPC, uma sub-rede e um ou mais grupos de segurança.

   e. (Opcional) Escolha uma chave de criptografia do AWS Key Management Service.

f. (Opcional) Adicione tags para a instância do bloco de anotações.

4. Escolha Criar caderno. Na página Notebooks (Blocos de anotações), escolha o ícone de atualização no canto superior direito e continue até que o Status seja `Ready`.
5. Marque a caixa de seleção ao lado do nome do novo bloco de anotações e escolha Open notebook (Abrir bloco de anotações).
6. Criar um novo bloco de anotações: na página jupyter, escolha New (Novo) e Sparkmagic (PySpark).

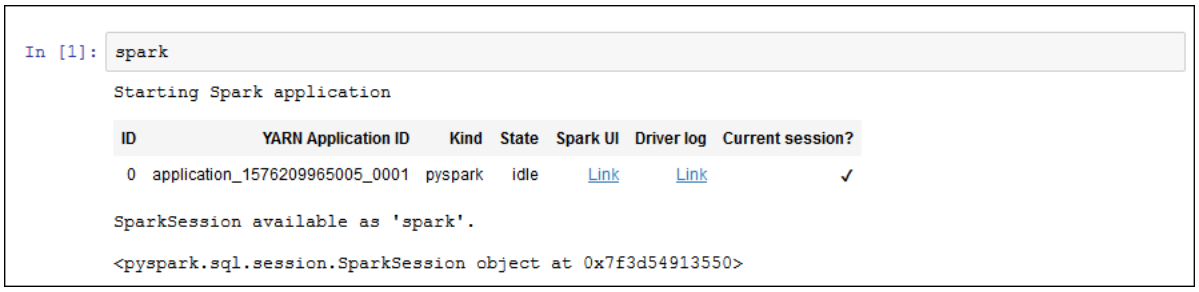
   Sua tela agora deve ser semelhante ao seguinte:



7. (Opcional) Na parte superior da página, escolha Untitled (Sem título) e dê um nome ao bloco de anotações.
8. Para iniciar um aplicativo do Spark, digite o seguinte comando no bloco de anotações e, na barra de ferramentas, escolha Run (Executar).

```
spark
```

   Após um pequeno atraso, você deve ver a seguinte resposta:



9. Criar um quadro dinâmico e executar uma consulta: copie, cole e execute o seguinte código, que gera a contagem e o esquema da tabela `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: usar um shell REPL com seu endpoint de desenvolvimento

No AWS Glue, você pode criar um endpoint de desenvolvimento e, em seguida, invocar um shell REPL (Read–Evaluate–Print Loop) para executar o código PySpark de forma incremental. Dessa forma, você poderá depurar interativamente seus scripts de ETL antes de implantá-los.

Para usar um REPL em um endpoint de desenvolvimento, você precisa ter autorização para SSH no endpoint.

1. No computador local, abra uma janela de terminal que possa executar comandos SSH e cole o comando SSH editado. Execute o comando.

   Pressupondo que você aceitou a versão 1.0 padrão do AWS Glue com Python 3 para o endpoint de desenvolvimento, a saída será assim:

```
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. Digite a instrução `print(spark.version)` para ver se o shell REPL está funcionando corretamente. Se a versão Spark for exibida, seu REPL está pronto para ser usado.

3. Agora você pode tentar executar o seguinte script simples, linha por linha, no 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: configurar um PyCharm Professional com um endpoint de desenvolvimento

Este tutorial mostra como conectar o IDE Python PyCharm Professional sendo executado em sua máquina local a um endpoint de desenvolvimento para que você possa executar, depurar e testar de forma interativa scripts de ETL (extração, transformação e carregamento) do AWS Glue antes de implantá-los. As instruções e capturas de tela no tutorial são baseadas no PyCharm Professional versão 2019.3.

Para se conectar a um endpoint de desenvolvimento de forma interativa, você precisa ter PyCharm Professional instalado. Você não pode fazer isso usando a edição gratuita.

> **Note**
>
> O tutorial usa o Amazon S3 como a fonte dos dados. Se você quiser usar uma fonte de dados JDBC, execute o endpoint de desenvolvimento em uma nuvem privada virtual (VPC). Para se conectar com SSH a um endpoint de desenvolvimento em uma VPC, você deve criar um túnel SSH. Este tutorial não inclui instruções para criar um túnel SSH. Para obter informações sobre como usar o SSH para se conectar a um endpoint de desenvolvimento em uma VPC, consulte Securely Connect to Linux Instances Running in a Private Amazon VPC no blog de segurança da AWS.

Tópicos

Conectar o PyCharm Professional a um endpoint de desenvolvimento

1. Crie um novo projeto Python puro no PyCharm, chamado `legislators`.
2. Crie um arquivo chamado `get_person_schema.py` no projeto com o seguinte conteúdo:

```
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()
```

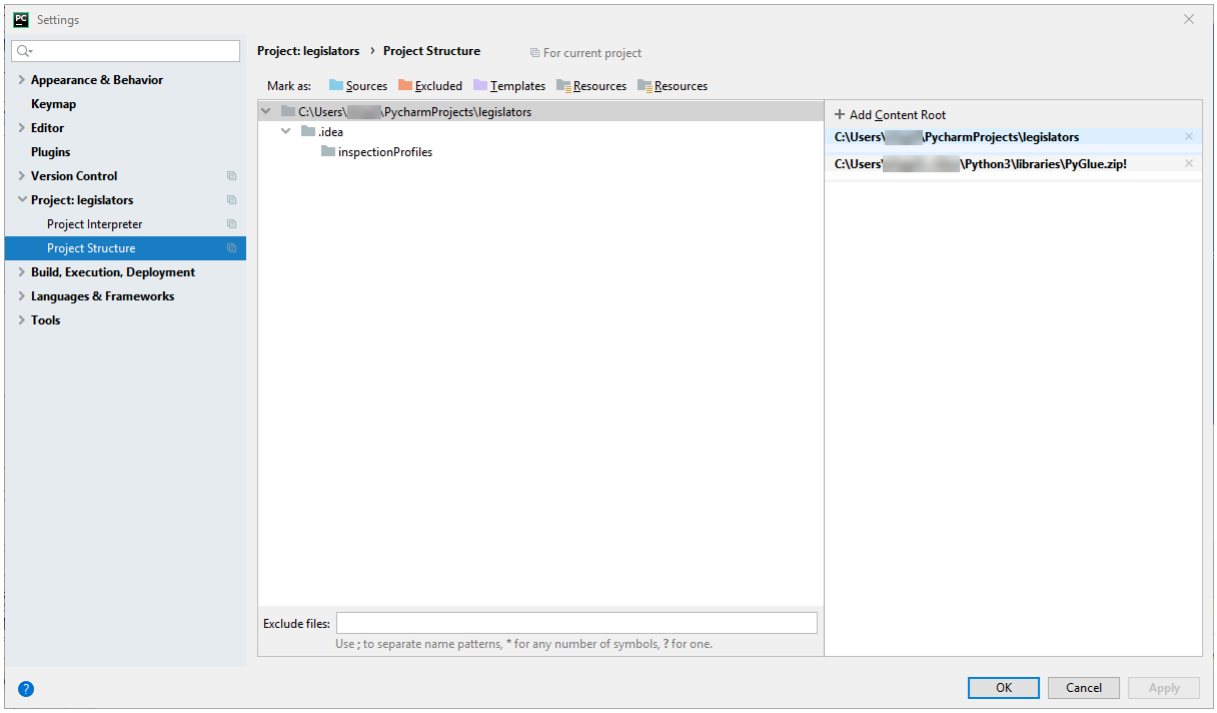
3. Execute um destes procedimentos:

   - Para o AWS Glue versão 0.9, faça download do arquivo da biblioteca Python do AWS Glue, `PyGlue.zip`, em `https://s3.amazonaws.com/aws-glue-jes-prod-us-east-1-assets/etl/python/PyGlue.zip` para um local conveniente na sua máquina local.
   - Para o AWS Glue versão 1.0 ou versões posteriores, faça download do arquivo da biblioteca Python do AWS Glue, `PyGlue.zip`, em `https://s3.amazonaws.com/aws-glue-jes-prod-us-east-1-assets/etl-1.0/python/PyGlue.zip` para um local conveniente na sua máquina local.

4. Adicione o `PyGlue.zip` como raiz de conteúdo do seu projeto no PyCharm:

   - No PyCharm, escolha File, Settings para abrir a caixa de diálogo Settings. Você também pode pressionar `Ctrl+Alt+S`.
   - Expanda o projeto `legislators` e escolha Project Structure. Em seguida, escolha + Add Content Root no painel direito.
   - Navegue até o local em que você salvou `PyGlue.zip`, selecione-o, e escolha Apply.

   A tela Settings deve ser semelhante à seguinte:



Mantenha a caixa de diálogo Settings aberta depois de selecionar Apply.

5. Configure as opções de implantação para fazer upload do script local no seu endpoint de desenvolvimento usando SFTP (esse recurso está disponível somente no PyCharm Professional):

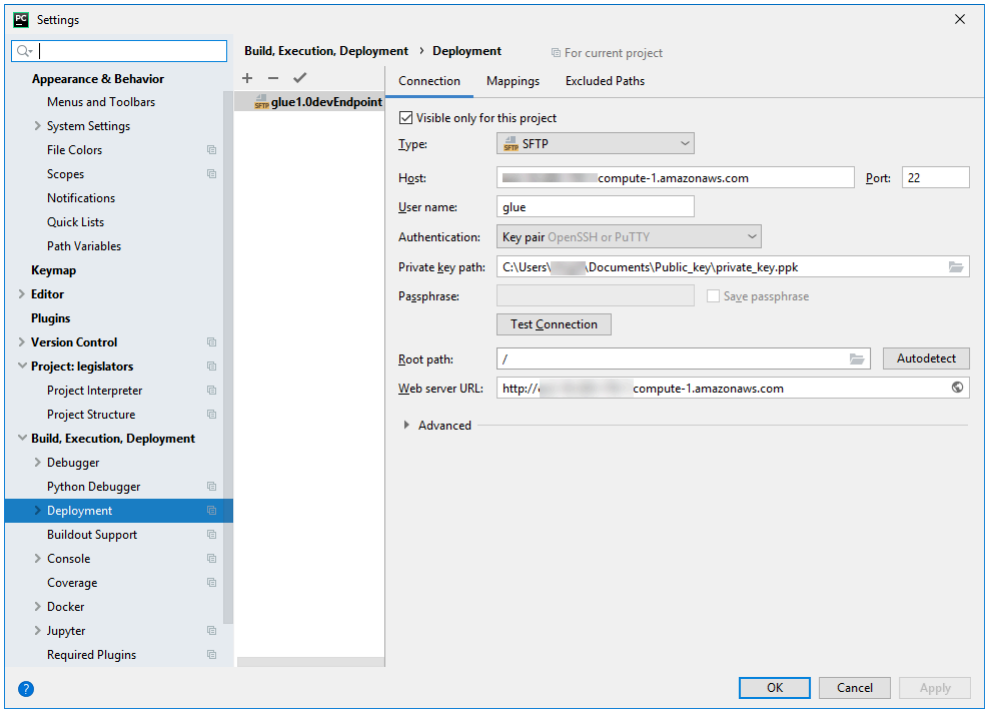
   - Na caixa de diálogo Settings, expanda a seção Build, Execution, Deployment. Escolha a subseção Deployment.
   - Escolha o ícone + na parte superior do painel central para adicionar um novo servidor. Defina o Type (Tipo) como SFTP e dê um nome a ele.
   - Defina o SFTP host (Host SFTP) como Public address (Endereço público) do endpoint de desenvolvimento, conforme listado na página de detalhes. Escolha o nome do endpoint de desenvolvimento no console do AWS Glue para exibir a página de detalhes. Para um endpoint de desenvolvimento em execução em uma VPC, defina o SFTP host (Host SFTP) como o endereço do host e a porta local do túnel SSH como o endpoint de desenvolvimento.
   - Defina User name como `glue`.
   - Defina Auth type como Key pair (OpenSSH or Putty). Defina Private key file (Arquivo de chave privada) navegando até o local em que o arquivo de chave privada do endpoint de desenvolvimento está localizado. Observe que PyCharm só é compatível com tipos de chave DSA, RSA e ECDSA OpenSSH, e não aceita chaves no formato privado de Putty. Você pode

usar uma versão atualizada de `ssh-keygen` para gerar um tipo de par de chaves aceito pelo PyCharm, usando uma sintaxe como a seguinte:

```
ssh-keygen -t rsa -f <key_file_name> -C "<your_email_address>"
```

- Escolha Test connection (Conexão de teste) e permita que a conexão seja testada. Se a conexão for concluída com sucesso, escolha Apply.

A tela Settings agora deve ser semelhante à seguinte:

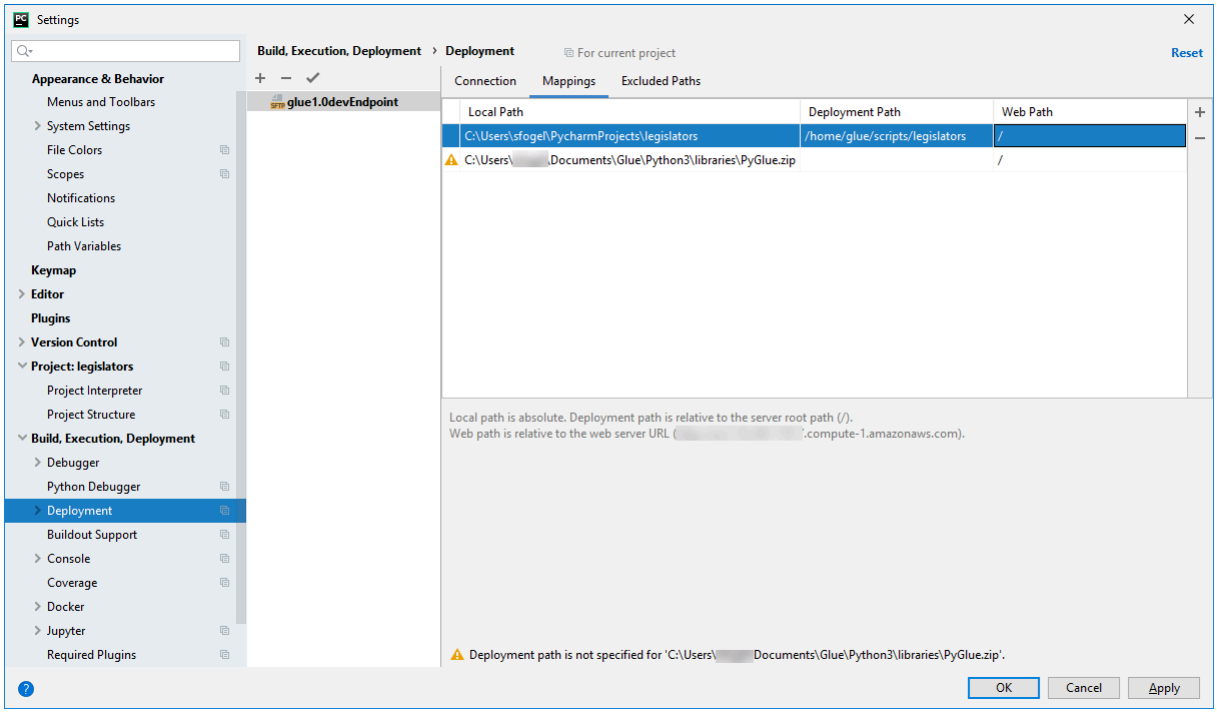


Novamente, mantenha a caixa de diálogo Settings aberta depois de selecionar Apply.

6. Mapeie o diretório local até um diretório remoto para implantação:

   - No painel direito da página Deployment, escolha a guia central na parte superior, nomeada Mappings.
   - Na coluna Deployment Path, insira um caminho abaixo de `/home/glue/scripts/` para implantação do caminho do seu projeto. Por exemplo: `/home/glue/scripts/legislators`.
   - Escolha Aplicar.

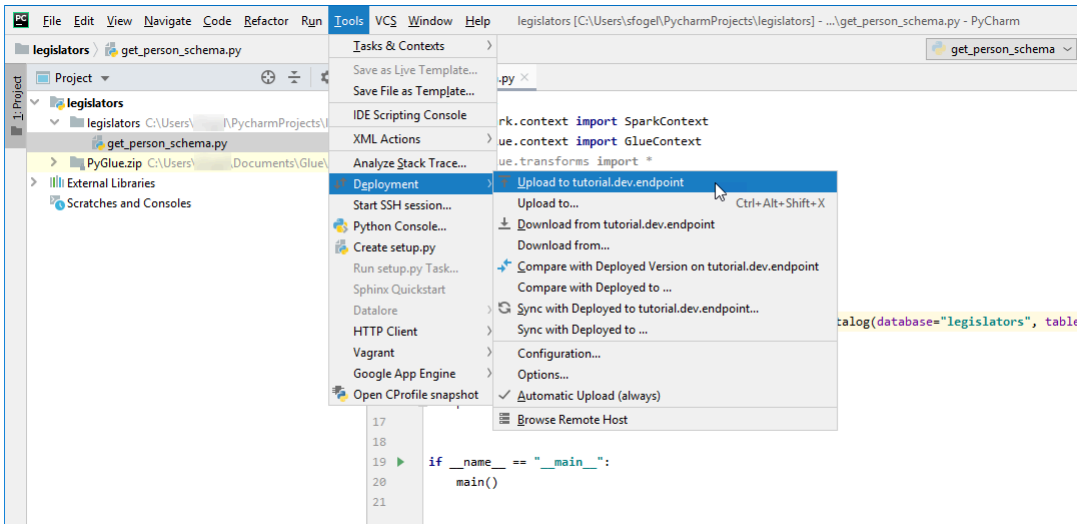
   A tela Settings agora deve ser semelhante à seguinte:



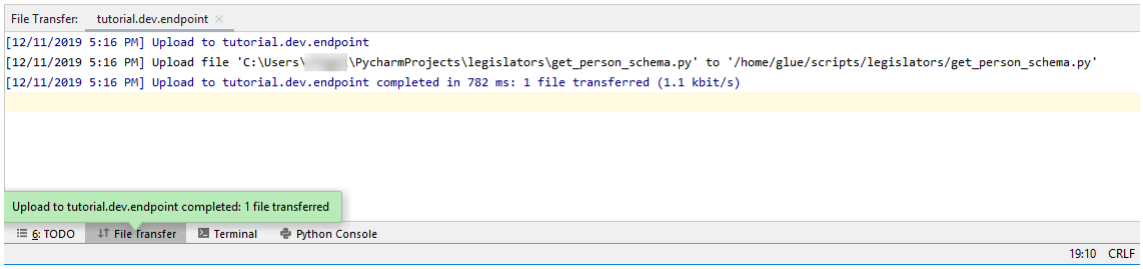
Escolha OK para fechar a caixa de diálogo Settings (Configurações).

Implantar o script no endpoint de desenvolvimento

1. Escolha Tools (Ferramentas), Deployment (Implantação) e o nome para o qual você configurou seu endpoint de desenvolvimento, conforme mostrado na seguinte imagem:



Depois que seu script for implantado, a parte inferior da tela exibirá o seguinte:



2. Na barra de menus, escolha Tools (Ferramentas), Deployment (Implantação), Automatic Upload (always) (Upload automático (sempre)). Verifique se uma marca de seleção aparece ao lado de Automatic Upload (always) (Upload automático (sempre)).

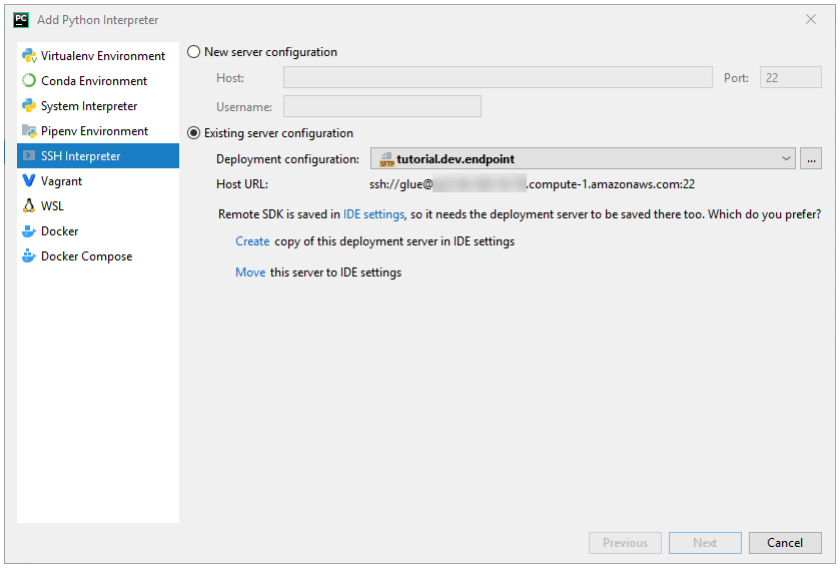
   Quando essa opção está habilitada, o PyCharm carrega automaticamente os arquivos alterados no endpoint de desenvolvimento.

Configurar um intérprete remoto

Configure o PyCharm para usar o intérprete Python no endpoint de desenvolvimento.

1. No menu File (Arquivo), escolha Settings (Configurações).
2. Expanda os legislators (legisladores) do projeto e escolha Project Interpreter (Intérprete do projeto).
3. Escolha o ícone de engrenagem ao lado da lista Project Interpreter (Intérprete do projeto) e escolha Add (Adicionar).
4. Na caixa de diálogo Add Python Interpreter (Adicionar intérprete Python), no painel esquerdo, escolha SSH Interpreter (Intérprete SSH).
5. Escolha Existing server configuration (Configuração de servidor existente) e, na lista Deployment configuration (Configuração de implantação), escolha sua configuração.

   Sua tela deve ser semelhante à seguinte imagem.



6. Escolha Move this server to IDE settings (Mover este servidor para configurações do IDE) e Next (Avançar).
7. No campo Interpreter (Intérprete), altere o caminho para `/usr/bin/gluepython` se você estiver usando Python 2 ou para `/usr/bin/gluepython3` se estiver usando Python 3. Em seguida, escolha Finish (Concluir).

Executar seu script no endpoint de desenvolvimento

Para executar o script:

- No painel esquerdo, clique com o botão direito do mouse no nome do arquivo e escolha Run '***<filename>***' (Executar <nome do arquivo>).

  Após uma série de mensagens, a saída final deve mostrar a contagem e o 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
```

Agora você está pronto para depurar seu script remotamente no endpoint de desenvolvimento.

Configuração avançada: compartilhar endpoints de desenvolvimento entre vários usuários

Esta seção explica como você pode aproveitar os endpoints de desenvolvimento com cadernos do SageMaker em casos de uso típicos a fim de compartilhar endpoints de desenvolvimento entre vários usuários.

Configuração de locação única

Em casos de uso de locatário único, para simplificar a experiência do desenvolvedor e para evitar contenção por recursos, é recomendável que cada desenvolvedor use seu próprio endpoint de desenvolvimento dimensionado para o projeto em que está trabalhando. Isso também simplifica as decisões relacionadas ao tipo de operador e à contagem de DPU, deixando-as à discrição do desenvolvedor e do projeto em que estão trabalhando.

Você não precisará cuidar da alocação de recursos, a menos que execute vários arquivos de caderno simultaneamente. Se você executar um código em vários arquivos de caderno ao mesmo tempo, várias sessões do Livy serão iniciadas simultaneamente. Para segregar configurações de cluster do Spark a fim de executar várias sessões do Livy ao mesmo tempo, você pode seguir as etapas que são introduzidas em casos de uso multilocatário.

Por exemplo, se o endpoint de desenvolvimento tiver dez operadores e o tipo de operador for `G.1X`, então você terá nove executores do Spark e todo o cluster terá 90G de memória de executor, uma vez que cada executor terá 10G de memória.

Independentemente do tipo de operador especificado, a alocação dinâmica de recursos do Spark será ativada. Se um conjunto de dados for grande o suficiente, o Spark pode alocar todos os executores em uma única sessão do Livy, já que `spark.dynamicAllocation.maxExecutors` não é definido por padrão. Isso significa que outras sessões do Livy no mesmo endpoint de desenvolvimento esperarão para iniciar novos executores. Se o conjunto de dados for pequeno, o Spark poderá alocar executores em várias sessões do Livy ao mesmo tempo.

> **Note**
>
> Para obter mais informações sobre como os recursos são alocados em diferentes casos de uso e como você define uma configuração para modificar o comportamento, consulte Configuração avançada: compartilhar endpoints de desenvolvimento entre vários usuários.

Configuração de locação múltipla

> **Note**
>
> Observe que os endpoints de desenvolvimento são destinados a emular o ambiente de ETL do AWS Glue como um ambiente de locatário único. Embora o uso de multilocatário seja possível, esse é um caso de uso avançado e é recomendado que a maioria dos usuários mantenha um padrão de locação única para cada endpoint de desenvolvimento.

Em casos de uso multilocatário, talvez seja necessário cuidar da alocação de recursos. O principal fator é o número de usuários simultâneos que usam um caderno Jupyter ao mesmo tempo. Se sua equipe estiver distribuída ao redor do mundo de forma a seguir um fluxo de trabalho de 24 horas por dia e houver apenas um usuário do Jupyter em cada fuso horário, então o número de usuários

simultâneos será apenas um e você não precisará se preocupar com a alocação de recursos. No entanto, se seu caderno for compartilhado entre vários usuários e cada um enviar código de forma ad-hoc, então você precisará considerar os pontos abaixo.

Para particionar recursos de cluster do Spark entre vários usuários, você pode usar as configurações do SparkMagic. Há duas maneiras diferentes de configurar o SparkMagic.

(A) Usar a diretiva %%configure -f

Se você quiser modificar a configuração por sessão do Livy a partir do caderno, poderá executar a diretiva `%%configure -f` no parágrafo do caderno.

Por exemplo, se você deseja executar a aplicação Spark em cinco executores, pode executar o seguinte comando no parágrafo do caderno.

```
%%configure -f
{"numExecutors":5}
```

Em seguida, você verá apenas cinco executores em execução para o trabalho na IU do Spark.

Recomendamos limitar o número máximo de executores para alocação dinâmica de recursos.

```
%%configure -f
{"conf":{"spark.dynamicAllocation.maxExecutors":"5"}}
```

(B) Modificar o arquivo Config do SparkMagic

O SparkMagic funciona com base na API do Livy. O SparkMagic cria sessões do Livy com configurações como: `driverMemory`, `driverCores`, `executorMemory`, `executorCores`, `numExecutors`, `conf` etc. Esses são os principais fatores que determinam a quantidade de recursos consumidos de todo o cluster do Spark. O SparkMagic permite que você forneça um arquivo de configuração para especificar os parâmetros que são enviados ao Livy. Você pode ver um arquivo de configuração de exemplo neste repositório do GitHub.

Se você quiser modificar a configuração em todas as sessões do Livy a partir de um caderno, poderá modificar `/home/ec2-user/.sparkmagic/config.json` para adicionar `session_config`.

Para modificar o arquivo de configuração em uma instância de cadernos do SageMaker, siga estas etapas.

1. Abra um caderno do SageMaker.
2. Abra o kernel do terminal.
3. Execute os seguintes comandos:

```
sh-4.2$ cd .sparkmagic
sh-4.2$ ls
config.json logs
sh-4.2$ sudo vim config.json
```

Por exemplo, você pode adicionar estas linhas a `/home/ec2-user/.sparkmagic/config.json` e reiniciar o kernel do Jupyter a partir do caderno.

```
  "session_configs": {
    "conf": {
      "spark.dynamicAllocation.maxExecutors":"5"
    }
  },
```

Diretrizes e práticas recomendadas

Para evitar esse tipo de conflito de recursos, você pode usar algumas abordagens básicas, como:

- Ter um cluster do Spark maior, aumentando `NumberOfWorkers` (escalonando horizontalmente) e atualizando `workerType`(escalonando verticalmente)
- Alocar menos recursos por usuário (menos recursos por sessão do Livy)

Sua abordagem dependerá do seu caso de uso. Se você tiver um endpoint de desenvolvimento maior e não houver uma grande quantidade de dados, a possibilidade de um conflito de recursos diminuirá significativamente porque o Spark pode alocar recursos com base em uma estratégia de alocação dinâmica.

Como descrito acima, o número máximo de executores do Spark é calculado automaticamente pela combinação de DPU (ou `NumberOfWorkers`) e tipo de operador. Cada aplicação do Spark inicia um driver e vários executores. Para calcular, você precisará de `NumberOfWorkers = NumberOfExecutors + 1`. A matriz abaixo explica quanta capacidade você precisa em seu endpoint de desenvolvimento com base no número de usuários simultâneos.

Número de usuários simultâneos de cadernos	Número de executores do Spark que você deseja alocar por usuário	NumberOfWorkers (número de operadores) total para seu endpoint de desenvolvimento
3	5	18
10	5	60
50	5	300

Se você quiser alocar menos recursos por usuário, `spark.dynamicAllocation.maxExecutors` (ou `numExecutors`) seria o parâmetro mais fácil de configurar como um parâmetro de sessão do Livy. Se você definir a configuração abaixo em `/home/ec2-user/.sparkmagic/config.json`, então o SparkMagic atribuirá um máximo de cinco executores por sessão do Livy. Isso ajudará a segregar recursos por sessão do Livy.

```
"session_configs": {
    "conf": {
      "spark.dynamicAllocation.maxExecutors":"5"
    }
  },
```

Suponha que haja um endpoint de desenvolvimento com 18 operadores (G.1X) e que haja três usuários de cadernos simultâneos. Se a configuração da sua sessão tiver `spark.dynamicAllocation.maxExecutors=5`, então cada usuário poderá fazer uso de um driver e cinco executores. Não haverá conflitos de recursos, mesmo quando você executar vários parágrafos do caderno ao mesmo tempo.

Trade-offs

Com esta configuração de sessão `"spark.dynamicAllocation.maxExecutors":"5"`, você poderá evitar erros de conflito de recursos e não precisará aguardar a alocação de recursos quando houver acessos simultâneos de usuários. No entanto, mesmo quando há muitos recursos livres (por exemplo, não há outros usuários simultâneos), o Spark não pode atribuir mais de cinco executores à sessão do Livy.

Outras observações

É uma boa prática interromper o kernel do Jupyter quando você parar de usar um caderno. Isso liberará recursos e outros usuários de cadernos poderão usar esses recursos imediatamente, sem aguardar a expiração do kernel (desligamento automático).

Problemas comuns

Mesmo ao seguir as diretrizes, você pode enfrentar certos problemas.

Sessão não encontrada

Ao tentar executar um parágrafo do caderno mesmo que sua sessão do Livy já tenha terminado, você verá a mensagem abaixo. Para ativar a sessão do Livy, você precisa reiniciar o kernel do Jupyter escolhendo Kernel > Restart (Reiniciar) no menu do Jupyter e, em seguida, executar o parágrafo do caderno novamente.

```
An error was encountered:
Invalid status code '404' from http://localhost:8998/sessions/13 with error payload:
 "Session '13' not found."
```

Não há recursos suficientes do YARN

Ao tentar executar um parágrafo do caderno mesmo que o cluster do Spark não tenha recursos suficientes para iniciar uma nova sessão do Livy, você verá a mensagem abaixo. Muitas vezes, você pode evitar esse problema seguindo as diretrizes, no entanto, ainda haverá a possibilidade de você encontrar esse problema. Para contornar o problema, você pode conferir se há sessões ativas e desnecessárias do Livy. Se houver sessões do Livy desnecessárias, você precisará terminá-las para liberar os recursos do cluster. Consulte a próxima seção para obter detalhes.

```
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.
```

Monitoramento e depuração

Esta seção descreve técnicas para monitorar recursos e sessões.

Monitorar e depurar a alocação de recursos do cluster

Você pode observar a interface do usuário do Spark para monitorar quantos recursos são alocados por sessão do Livy e quais são as configurações efetivas do Spark no trabalho. Para habilitar a interface do usuário do Spark, consulte Habilitar a interface do usuário Web do Apache Spark para endpoints de desenvolvimento.

(Opcional) Se você precisar de uma visualização em tempo real da interface do usuário do Spark, poderá configurar um túnel SSH para o servidor de histórico do Spark em execução no cluster do Spark.

```
ssh -i <private-key.pem> -N -L 8157:<development endpoint public address>:18080
 glue@<development endpoint public address>
```

Você pode abrir http://localhost:8157 no navegador para visualizar a interface do usuário do Spark localmente.

Liberar sessões do Livy desnecessárias

Reveja estes procedimentos para encerrar quaisquer sessões desnecessárias do Livy a partir de um caderno ou de um cluster do Spark.

(a). Terminar sessões do Livy a partir de um caderno

Você pode desligar o kernel em um caderno Jupyter para terminar sessões do Livy desnecessárias.

(b). Terminar sessões do Livy a partir de um cluster do Spark

Se houver sessões do Livy desnecessárias que ainda estejam em execução, você pode terminá-las no cluster do Spark.

Como pré-requisito para executar esse procedimento, você precisa configurar sua chave SSH pública para o seu endpoint de desenvolvimento.

Para fazer login no cluster do Spark, execute o seguinte comando:

```
$ ssh -i <private-key.pem> glue@<development endpoint public address>
```

Você pode executar o seguinte comando para ver as sessões do Livy ativas:

```
$ 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
```

Em seguida, você pode desligar a sessão do Livy com o seguinte 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
```

Gerenciar cadernos

> **Note**
>
> Os endpoints de desenvolvimento só são compatíveis com as versões do AWS Glue anteriores à 2.0. Para um ambiente interativo onde você possa criar e testar scripts de ETL, use Cadernos no AWS Glue Studio.

Um caderno permite o desenvolvimento e teste interativos de seus scripts de ETL (extrair, transformar e carregar) em um endpoint de desenvolvimento. O AWS Glue fornece uma interface para cadernos Jupyter do SageMaker. Com o AWS Glue, você cria e gerencia cadernos do SageMaker. Você também pode abrir cadernos do SageMaker no console do AWS Glue.

Além disso, você pode usar o Apache Spark com o SageMaker em endpoints de desenvolvimento do AWS Glue que oferecem suporte ao SageMaker (mas não a trabalhos de ETL do AWS Glue). O SageMaker Spark é uma biblioteca de código aberto do Apache Spark para SageMaker. Para obter mais informações, consulte Uso do Apache Spark com o Amazon SageMaker.

Important

O gerenciamento de cadernos do SageMaker com endpoints de desenvolvimento do AWS Glue está disponível nas seguintes regiões da AWS:

Região	Código
Leste dos EUA (Ohio)	`us-east-2`
Leste dos EUA (N. da Virgínia)	`us-east-1`
Oeste dos EUA (N. da Califórnia)	`us-west-1`
Oeste dos EUA (Oregon)	`us-west-2`
Ásia-Pacífico (Tóquio)	`ap-northeast-1`
Ásia-Pacífico (Seul)	`ap-northeast-2`
Ásia-Pacífico (Mumbai)	`ap-south-1`
Ásia-Pacífico (Singapura)	`ap-southeast-1`
Ásia-Pacífico (Sydney)	`ap-southeast-2`
Canadá (Central)	`ca-central-1`
Europa (Frankfurt)	`eu-central-1`
Europa (Irlanda)	`eu-west-1`
Europa (Londres)	`eu-west-2`

Criar trabalhos ETL visuais com o AWS Glue Studio

Um trabalho do AWS Glue encapsula um script que se conecta aos dados de origem, os processa e, depois, os grava no destino de dados. Normalmente, um trabalho executa scripts de extração, transformação e carga (ETL). Os trabalhos podem executar scripts desenhados para os ambientes de runtime do Apache Spark e do Ray. Os trabalhos também podem executar scripts Python de uso geral (trabalhos do shell do Python). Os acionadores do AWS Glue podem iniciar trabalhos com base em uma programação ou um evento, ou sob demanda. É possível monitorar trabalhos para entender as métricas do runtime, status de conclusão, duração e hora de início.

Você pode usar scripts gerados pelo AWS Glue ou fornecer os seus próprios scripts. Com um esquema de fonte e um local de destino ou esquema, o gerador de códigos do AWS Glue Studio pode criar automaticamente um script da API Apache Spark (PySpark). Você pode usar esse script como ponto de partida e editá-lo para atingir seus objetivos.

O AWS Glue pode gravar arquivos de saída em vários formatos de dados. Cada tipo de trabalho pode ser compatível com diferentes formatos de saída. Para alguns formatos de dados, é possível gravar formatos de compressão comuns.

Fazer login no console AWS Glue

Um trabalho no AWS Glue consiste na lógica de negócios que realiza o trabalho de extração, transformação e carregamento (ETL). Você pode criar trabalhos na seção ETL do console do AWS Glue.

Para visualizar trabalhos existentes, faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/. Em seguida, escolha a guia Jobs no AWS Glue. A lista Jobs exibe o local do script associado a cada trabalho, a hora em que o trabalho foi modificado pela última vez e a opção atual do marcador de trabalho.

Ao criar um novo trabalho, ou depois de ter salvo seu trabalho, você pode usar o AWS Glue Studio para modificar seus trabalhos de ETL. Você pode fazer isso editando os nós no editor visual ou editando o script de trabalho no modo de desenvolvedor. Você também pode adicionar e remover nós no editor visual para criar trabalhos de ETL mais complicados.

Próximas etapas para criar um trabalho no AWS Glue Studio

Use o editor de trabalhos visual para configurar nós para seu trabalho. Cada nó representa uma ação, como ler dados do local de origem ou aplicar uma transformação aos dados. Cada nó adicionado ao trabalho tem propriedades que fornecem informações sobre o local dos dados ou a transformação.

As próximas etapas para criar e gerenciar seus trabalhos são:

- Visual ETL com AWS Glue Studio
- Visualizar o script de trabalho
- Modificar as propriedades do trabalho
- Salvar o trabalho
- Iniciar uma execução de trabalho
- Exibir informações para execuções de trabalho recentes
- Acessar o painel de monitoramento de trabalhos

Visual ETL com AWS Glue Studio

Você pode usar a interface visual simples no AWS Glue Studio para criar seus trabalhos de ETL. Você pode usar a página Jobs (Trabalhos) para criar novos trabalhos. Você também pode usar um editor de scripts ou bloco e anotações para trabalhar diretamente com código no script de trabalho de ETL do AWS Glue Studio.

Na página Jobs (Trabalhos), você também pode ver todos os trabalhos que criou com o AWS Glue Studio ou o AWS Glue. Você pode exibir, gerenciar e executar seus trabalhos nessa página.

Veja também o tutorial do blog em outro exemplo de como criar trabalhos de ETL com o AWS Glue Studio.

Iniciar trabalhos no AWS Glue Studio

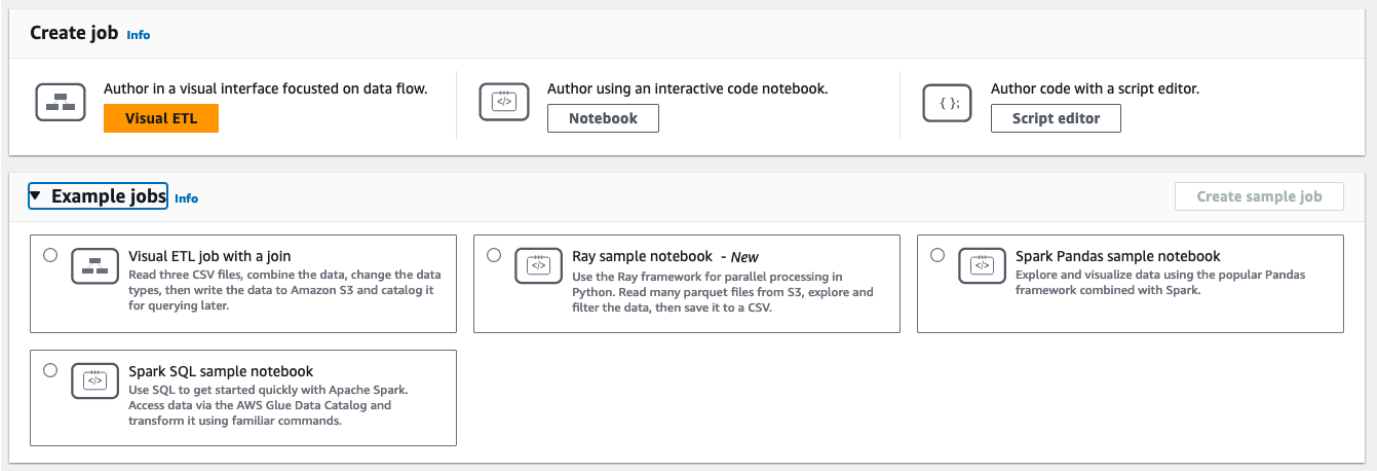
O AWS Glue permite criar um trabalho por meio de uma interface visual, um caderno de código interativo ou com um editor de scripts. Você pode iniciar um trabalho clicando em qualquer uma das opções ou criar um novo trabalho com base em um trabalho de exemplo.

Exemplos de trabalhos de criação de trabalho com a ferramenta de sua escolha. Por exemplo, exemplos de trabalhos permitem que você crie um trabalho ETL visual que une arquivos CSV em

uma tabela de catálogo, crie um trabalho em um caderno de código interativo com o AWS Glue para Ray ou AWS Glue para Spark ao trabalhar com pandas ou crie uma tarefa em um caderno de código interativo com o SparkSQL.

Criando um trabalho no AWS Glue Studio do zero

1. Faça login no AWS Management Console e abra o AWS Glue Studio console em https://console.aws.amazon.com/gluestudio/.
2. No painel de navegação, escolha Trabalhos.
3. Na seção Criar trabalho, escolha uma opção de configuração para o seu trabalho.



Opções para criar um trabalho do zero:

- Visual ETL: criar em uma interface visual focada no fluxo de dados
- Criar usando um caderno de código interativo: criar trabalhos interativamente em uma interface de caderno baseada em cadernos Jupyter

  Quando seleciona esta opção, você deve fornecer informações adicionais antes de criar uma sessão de criação de caderno. Para obter mais informações sobre como especificar essas informações, consulte Conceitos básicos de cadernos no AWS Glue Studio.
- Editor de scripts Spark: para aqueles familiarizados com programação e escrita de scripts de ETL, escolha essa opção para criar um novo trabalho de ETL do Spark. Escolha o mecanismo (Python shell, Ray, Spark (Python) ou Spark (Scala). Em seguida, escolha Começar do zero ou Carregar script. Carregar um script existente a partir de um arquivo local. Se você optar por usar o editor de scripts, não será possível usar o editor de trabalhos visual para criar ou editar seu trabalho.

Um trabalho do Spark é executado em um ambiente Apache Spark gerenciado pelo AWS Glue. Por padrão, novos scripts são codificados em Python. Para escrever um novo script em Scala, consulte Criar e editar scripts em Scala no AWS Glue Studio.

Criar um trabalho no AWS Glue Studio partir de um trabalho de exemplo

Você pode optar por criar um trabalho a partir de um trabalho de exemplo. Na seção Trabalhos de exemplo, escolha um trabalho de exemplo e, em seguida, escolha Criar trabalho de exemplo. A criação de um trabalho de exemplo a partir de uma das opções fornece um modelo rápido com o qual você pode trabalhar.

1. Faça login no AWS Management Console e abra o AWS Glue Studio console em https://console.aws.amazon.com/gluestudio/.
2. No painel de navegação, escolha Trabalhos.
3. Selecione uma opção para criar um trabalho a partir de um trabalho de exemplo:
   - Trabalho de ETL visual para unir várias fontes: ler três arquivos CSV, combinar os dados, alterar os tipos de dados e, em seguida, gravar os dados no Amazon S3 e catalogá-los para consulta posterior.
   - Notebook Spark usando Pandas: explorar e visualizar dados usando a popular estrutura do Panda combinada com o Spark.
   - Caderno Spark usando SQL: usar SQL para começar a usar rapidamente o Apache Spark. Acessar dados por meio do catálogo de dados do AWS Glue e transformá-los usando comandos familiares.
4. Escolha Criar trabalho de exemplo.

Recursos do editor de trabalhos

O editor de trabalhos fornece os seguintes recursos para criar e editar trabalhos.

- Um diagrama visual do seu trabalho, com um nó para cada tarefa de trabalho: nós de origem de dados para lê-los, transformar nós para modificá-los, nós de destino de dados para gravá-los.

  Você pode exibir e configurar as propriedades de cada nó no diagrama de trabalho. Você também pode exibir os dados de esquema e de exemplo de cada nó no diagrama de trabalho. Esses

recursos ajudam você a verificar se seu trabalho está modificando e transformando os dados da maneira correta, sem ter que executar o trabalho.

- Uma guia de exibição e edição de scripts, onde você pode modificar o código gerado para seu trabalho.
- Uma guia de detalhes do trabalho, onde você pode configurar uma variedade de configurações para personalizar o ambiente no qual seu trabalho de ETL do AWS Glue é executado.
- Uma guia de execuções, onde você pode exibir as execuções atuais e anteriores do trabalho, visualizar o status da execução do trabalho e acessar os logs da execução do trabalho.
- Uma guia Qualidade de dados, na qual você pode aplicar regras de qualidade de dados ao seu trabalho.
- Uma guia de programações, na qual você pode configurar o momento de início de seu trabalho ou configurar execuções recorrentes de um trabalho.
- Uma guia Controle de versão, na qual você pode configurar um serviço Git para usar com seu trabalho.

Usar previsualizações de esquema no editor de trabalhos visual

Ao criar ou editar seu trabalho, você pode usar a guia Output schema (Esquema de saída) para exibir o esquema dos dados.

Antes que você possa ver o esquema, o editor de trabalhos precisa de permissões para acessar a origem dos dados. Você pode especificar uma função do IAM para um nó na guia Job details (Detalhes do trabalho) do editor ou na guia Output schema (Esquema de saída). Se a função do IAM tiver todas as permissões necessárias para acessar a origem dos dados, você poderá visualizar o esquema na guia Output schema (Esquema de saída) de um nó.

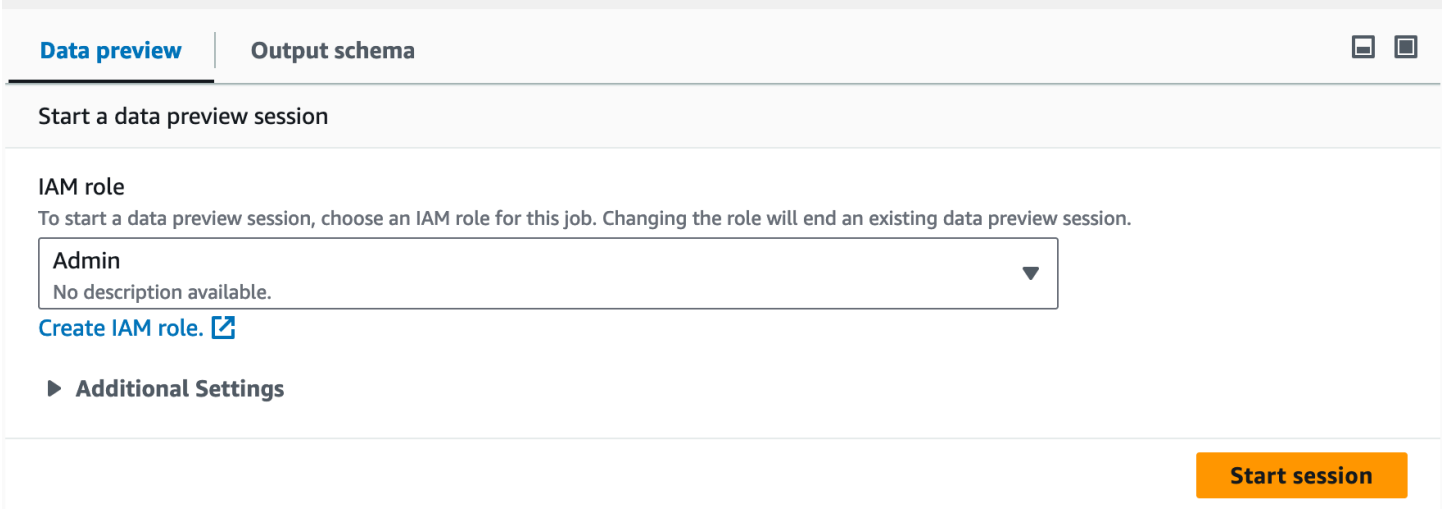
Usar previsualizações de dados no editor de trabalhos visual

As pré-visualizações de dados ajudam você a criar e testar seu trabalho usando um exemplo dos seus dados sem precisar executá-lo repetidamente. Ao usar a pré-visualização de dados, você pode:

- Testar um perfil do IAM para garantir que ele tenha acesso a suas fontes ou destinos de dados.
- Verificar se a transformação está modificando os dados da maneira pretendida. Por exemplo, se você usar uma transformação de filtro, pode certificar-se de que o filtro está selecionando o subconjunto correto de dados.

- Verifique seus dados. Se o conjunto de dados contiver colunas com valores de vários tipos, a visualização de dados mostrará uma lista de tuplas para elas. Cada tupla contém o tipo de dado e seu valor.

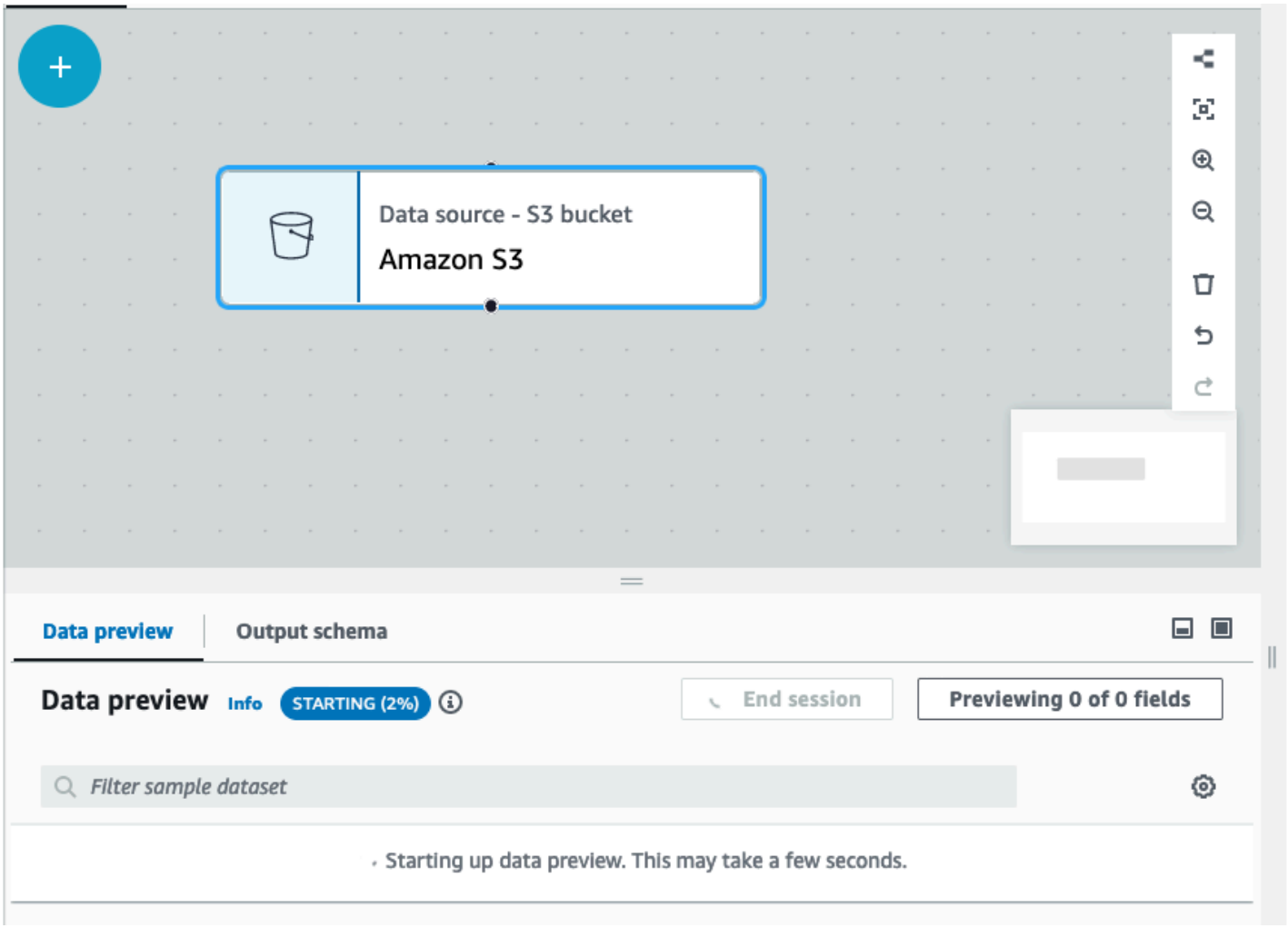
Ao criar ou editar seu trabalho, você pode usar a guia Visualização de dados abaixo da tela do trabalho para visualizar uma amostra dos seus dados. Uma nova sessão de pré-visualização de dados será iniciada automaticamente quando o perfil já estiver configurado no trabalho ou um perfil do IAM padrão tiver sido configurado na conta. Se um perfil não tiver sido configurado anteriormente, você poderá iniciar uma sessão selecionando o perfil.



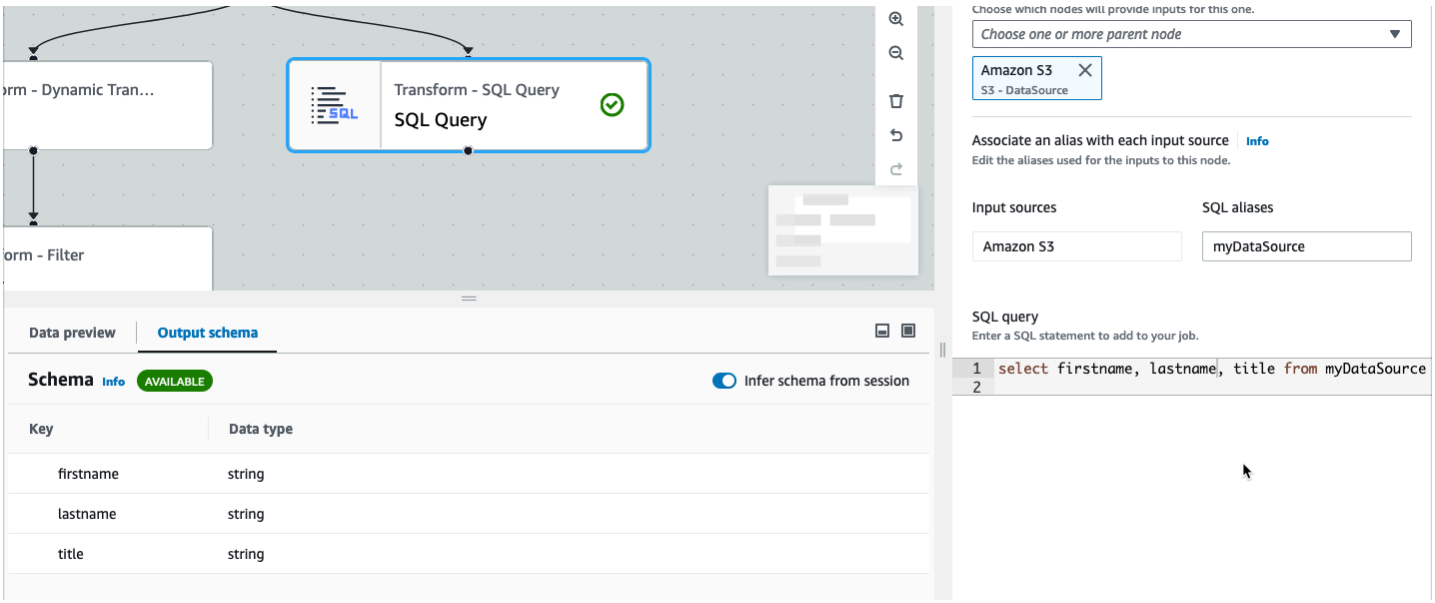
> **Note**
>
> O perfil que você escolher para a sessão de pré-visualização de dados também será usado para o trabalho.

É possível ver o status e o progresso da sua sessão, bem como os detalhes da sessão, clicando no ícone de informações.

Quando a sessão estiver pronta, o AWS Glue Studio carregará os dados para o nó selecionado. É possível visualizar o % concluído à medida que a sessão avança.



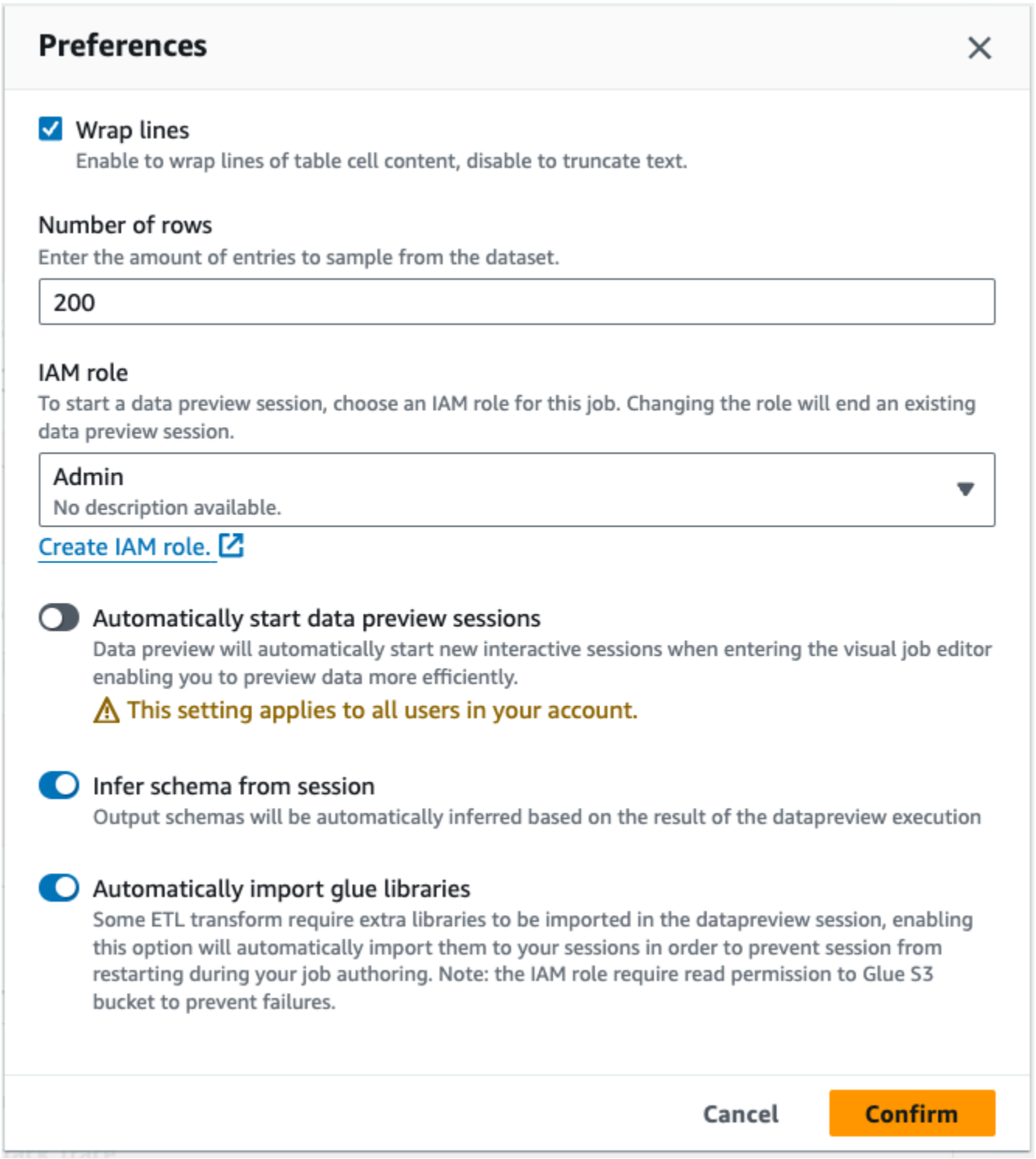
À medida que você criar seu trabalho visual, o AWS Glue Studio atualizará automaticamente o esquema do nó selecionado quando você alternar Inferir esquema da sessão na guia Esquema de saída.



Para configurar suas preferências de pré-visualização de dados:

Escolha o ícone de configurações (um símbolo de engrenagem) para configurar suas preferências de previsualizações de dados. Essas configurações se aplicam a todos os nós no diagrama de trabalho. É possível:

- Optar por quebrar o texto de uma linha para a próxima. Essa opção é habilitada por padrão
- Alterar o número de linhas (o padrão é 200)
- Escolher ou, se necessário, criar um perfil do IAM
- Optar por iniciar automaticamente uma nova sessão ao criar um trabalho. Isso fornece uma nova sessão interativa ao criar trabalhos. Essa configuração é aplicável no nível da conta. Uma vez definida, a configuração será aplicada a todos os usuários da sua conta ao editar qualquer trabalho.
- Escolher inferir automaticamente o esquema. Os esquemas de saída serão automaticamente inferidos para o nó selecionado
- Optar por importar automaticamente bibliotecas do AWS Glue. Isso é útil porque evitará que a pré-visualização de dados reinicie novas sessões ao adicionar novas transformações que exijam a reinicialização da sessão

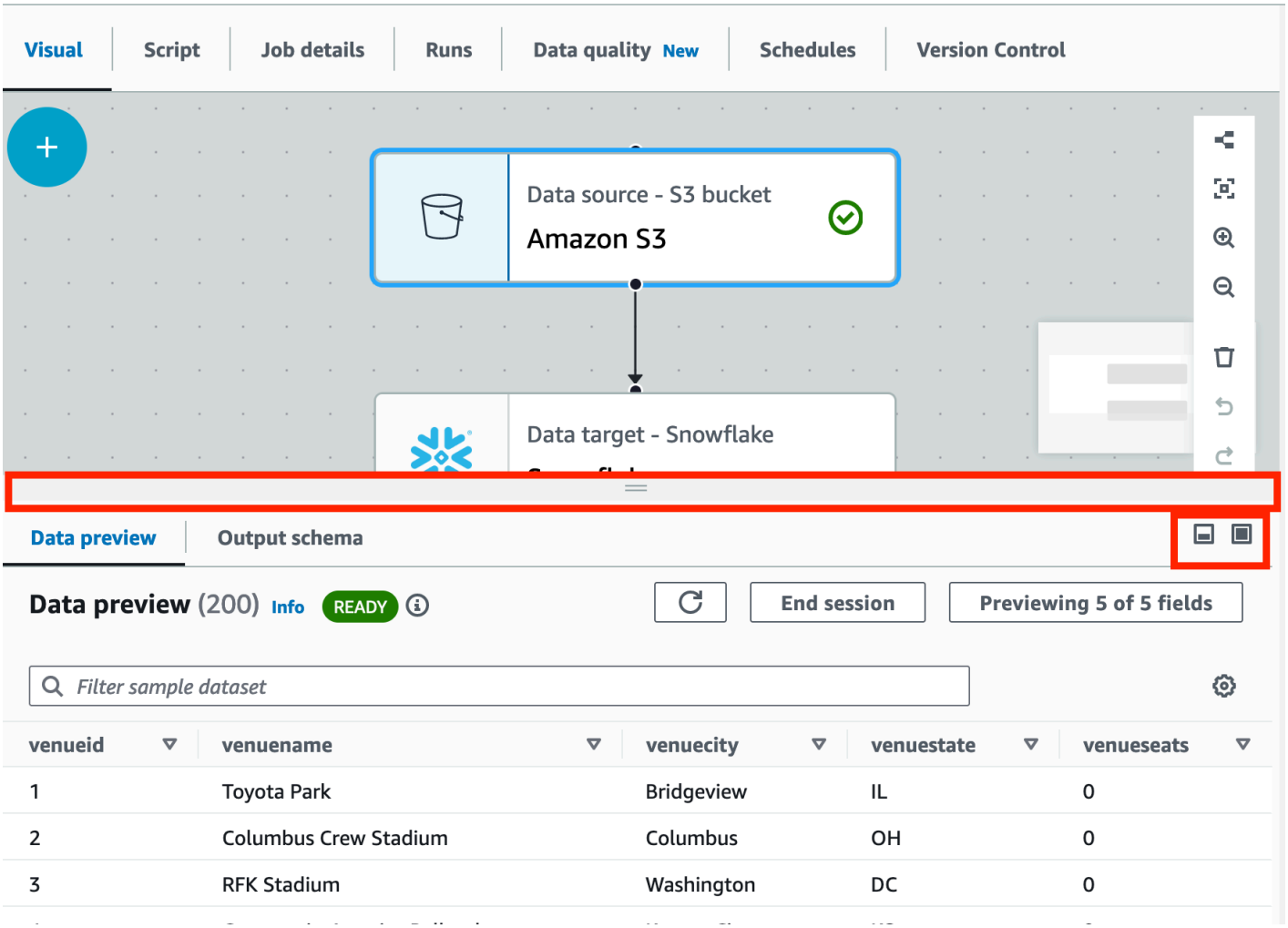


Os recursos adicionais incluem a capacidade de:

- Selecione o botão Previewing x of y fields (Previsualização de x de y campos) para selecionar quais colunas (campos) serão previsualizadas. Quando você previsualiza seus dados usando as

configurações padrão, o editor de trabalhos mostra as cinco primeiras colunas do seu conjunto de dados. Você pode alterar essa opção para mostrar todas ou nenhuma (não recomendado).

- Role pela janela de pré-visualização de dados, tanto na horizontal quanto na vertical.
- Use o botão maximizar para expandir a guia Visualização de dados para sobrepor o gráfico do trabalho e visualizar melhor os dados e as estruturas de dados. Da mesma forma, use o botão minimizar para minimizar a guia Visualização de dados. Você também pode pegar o painel da alça e arrastar para cima para expandir a guia Visualização de dados.



- Use Encerrar sessão para interromper a visualização dos dados. Ao interromper a sessão, você pode escolher um novo perfil do IAM e definir configurações adicionais (como ativar ou desativar configurações) para iniciar automaticamente uma nova sessão, inferir um esquema ou importar AWS Glue bibliotecas e iniciar a sessão novamente.

Restrições ao usar previsualizações de dados

Ao usar previsualizações de dados, você pode encontrar as seguintes restrições ou limitações.

- Na primeira vez que você escolhe a guia Data preview (Previsualização de dados), deve escolher uma função do IAM. Essa função deve ter as permissões exigidas para acessar os dados e outros recursos necessários para criar as previsualizações de dados.
- Depois de fornecer uma função do IAM, leva um tempo até que os dados estejam disponíveis para visualização. Para conjuntos de dados com menos de 1 GB de dados, pode levar até um minuto. Se você tiver um conjunto de dados grande, deverá usar partições para melhorar o tempo de carregamento. Carregar dados diretamente no Amazon S3 apresenta a melhor performance.
- Se você tiver um conjunto de dados muito grande e levar mais de 15 minutos para consultar os dados para a previsualização, a solicitação excederá o tempo limite. As pré-visualizações de dados têm um tempo limite de inatividade de 30 minutos. Para aliviar isso, reduza o tamanho do conjunto de dados para usar previsualizações de dados.
- Por padrão, você vê as 50 primeiras colunas na guia Data preview (Previsualização de dados). Se as colunas não tiverem valores de dados, você receberá uma mensagem informando que não há dados para exibir. Você pode aumentar o número de linhas de amostra ou de colunas diferentes selecionadas para ver os valores dos dados.
- No momento, as previsualizações de dados não são suportadas para origens de dados de transmissão ou para origens de dados que usam conectores personalizados.
- Erros em um nó afetam todo o trabalho. Se qualquer nó tiver um erro nas previsualizações de dados, ele aparecerá em todos os nós até que você o corrija.
- Se você alterar uma origem de dados para o trabalho, os nós filhos dessa origem de dados talvez precisem ser atualizados para corresponder ao novo esquema. Por exemplo, se você tiver um nó ApplyMapping que modifica uma coluna e ela não existir na origem de dados de substituição, será necessário atualizar o nó de transformação ApplyMapping.
- Se você exibir a guia Data preview (Previsualização de dados) para um nó de transformação de consulta SQL e tal consulta usar um nome de campo incorreto, a guia Data preview (Previsualização de dados) exibirá um erro.

Geração de código de script

Quando você usa o editor visual para criar um trabalho, o código ETL é gerado automaticamente para você. O AWS Glue Studio cria um script de trabalho funcional e completo e o salva em um local do Amazon S3.

Existem duas formas de código geradas pelo AWS Glue Studio: a versão original, ou clássica, e uma versão mais nova e racionalizada. Por padrão, o novo gerador de código é usado para criar o script de trabalho. Você pode gerar um script de trabalho usando o gerador de código clássico na guia Script escolhendo o botão Generate classic script (Gerar script clássico).

Algumas das diferenças na nova versão do código gerado incluem:

- Blocos de comentários grandes não são mais adicionados ao script
- As estruturas de saída no código usam o nome do nó especificado no editor visual. No script clássico, as estruturas de saída são simplesmente nomeadas `DataSource0`, `DataSource1`, `Transform0`, `Transform1`, `DataSink0`, `DataSink1` e assim por diante.
- Comandos longos são divididos em várias linhas para eliminar a necessidade de rolar pela página para ver o comando inteiro.

Novos recursos no AWS Glue Studio exigem a nova versão da geração de código e não funcionarão com o script de código clássico. Você será solicitado a atualizar esses trabalhos ao tentar executá-los.

Editar nós de transformação de dados gerenciados pelo AWS Glue

O AWS Glue Studio fornece dois tipos de transformação:

- Transformações nativas do AWS Glue, disponíveis para todos os usuários e gerenciadas pelo AWS Glue.
- Transformações visuais personalizadas, permitem que você carregue suas próprias transformações para usar no AWS Glue Studio

Nós de transformação de dados gerenciados pelo AWS Glue

O AWS Glue Studio fornece um conjunto de transformações integradas que você pode usar para processar seus dados. Seus dados passam de um nó no diagrama de trabalho para outro em uma estrutura de dados chamada `DynamicFrame`, que é uma extensão para um `DataFrame` do Apache Spark SQL.

No diagrama previamente preenchido de um trabalho, entre os nós de fonte de dados e de destino de dados está o nó de transformação Change Schema. Você pode configurar esse nó de transformação para modificar seus dados ou pode usar transformações adicionais.

As transformações integradas a seguir estão disponíveis no AWS Glue Studio:

- ChangeSchema: mapear chaves de propriedade de dados na fonte de dados para chaves de propriedade de dados no destino dos dados. Você pode renomear chaves, modificar os tipos de dados para chaves e escolher quais chaves remover do conjunto de dados.
- SelectFields (Selecionar campos): escolha as chaves de propriedade de dados que você deseja manter.
- DropFields (Descartar campos): escolha as chaves de propriedade de dados que você deseja descartar.
- RenameField (Renomear campo): renomeie uma única chave de propriedade de dados.
- Spigot (Torneira): grave amostras dos dados em um bucket do Amazon S3.
- Join (Unir): una dois conjuntos de dados em um só, usando uma frase de comparação nas chaves de propriedade de dados especificadas. Você pode usar junção inner (interna), outer (externa), left (à esquerda), right (à direita), left semi (semi à esquerda) e left anti (anti à esquerda).
- Union: combinar linhas de mais de uma fonte de dados que tenham o mesmo esquema.
- SplitFields (Dividir campos): divida chaves de propriedade de dados em dois `DynamicFrames`. A saída é uma coleção de `DynamicFrames`: um com chaves de propriedade de dados selecionadas e outro com as chaves de propriedade de dados restantes.
- SelectFromCollection (Selecionar na coleção): escolha um `DynamicFrame` a partir de uma coleção de `DynamicFrames`. A saída é o `DynamicFrame` selecionado.
- FillMissingValues (Preencher valores ausentes): localize registros no conjunto de dados que têm valores ausentes e adicione um novo campo com um valor determinado por imputação.
- Filter (Filtro): divida um conjunto de dados em dois, com base em uma condição de filtro.
- Drop Null Fields (Descartar campos nulos): remove colunas do conjunto de dados se todos os valores na coluna forem "null" (nulo).
- Drop Duplicates (Eliminar duplicatas): remove linhas da sua fonte de dados escolhendo combinar linhas inteiras ou especificar chaves.
- SQL: insira o código SparkSQL em um campo de entrada de texto para usar uma consulta SQL para transformar os dados. A saída é um único `DynamicFrame`.
- Agregar: executa um cálculo (como média, soma, mínimo, máximo) em campos e linhas selecionados, e cria um novo campo com os valores recém-calculados.
- Nivelar: extraia campos dentro de structs para campos de nível superior.
- UUID: adicione uma coluna com um identificador universal exclusivo para cada linha.

- Identificador: adicione uma coluna com um identificador numérico para cada linha.
- Em timestamp: converta uma coluna em tipo timestamp.
- Formatar timestamp: converta uma coluna de timestamp em uma string formatada.
- Transformação de roteador condicional: aplique várias condições aos dados de entrada. Cada linha dos dados de entrada avaliada por uma condição de filtro de grupo e processada no grupo correspondente.
- Transformação Concatenate Columns: Criar uma nova coluna de strings usando os valores de outras colunas com um espaçador opcional.
- Transformação Split String: permite que você divida uma string em uma matriz de tokens usando uma expressão regular para definir como a divisão é feita.
- Transformação Array To Columns: extrair alguns ou todos os elementos de uma coluna do tipo matriz para novas colunas.
- Transformação Add Current Timestamp: permite marcar as linhas com a hora em que os dados foram processados. Isso é útil para fins de auditoria ou para rastrear a latência no pipeline de dados.
- Transformação Pivot Rows to Columns: agregar uma coluna numérica girando valores únicos nas colunas selecionadas que se tornam novas colunas. Se várias colunas forem selecionadas, os valores serão concatenados para nomear as novas colunas.
- Transformação Unpivot Columns To Rows: converter colunas em valores de novas colunas, gerando uma linha para cada valor único.
- Transformação Autobalance Processing: redistribuir os dados entre os operadores para melhorar a performance. Isso ajuda nos casos em que os dados estão desbalanceados ou, como vêm da fonte, não permitem processamento paralelo suficiente.
- Transformação Derived Column: definir uma nova coluna com base em uma fórmula matemática ou expressão SQL na qual você pode usar outras colunas nos dados, além de constantes e literais.
- Transformação Lookup: adicionar colunas de uma tabela de catálogo definida quando as chaves correspondem às colunas de pesquisa nos dados.
- Transformação Explode Array Map Into Rows: extrair valores de uma estrutura aninhada em linhas individuais que são mais fáceis de manipular.
- Transformação Record matching: invocar uma transformação de classificação de dados de machine learning Record Matching existente.

- Transformação Remove null rows: remover do conjunto de dados as linhas que têm todas as colunas nulas ou vazias.
- Transformação Parse JSON comumn: analisar uma coluna de strings contendo dados JSON e a converter em uma estrutura ou coluna de matriz, dependendo se o JSON for um objeto ou uma matriz, respectivamente.
- Transformação Extract JSON path: extrair novas colunas de uma coluna de strings JSON.
- EssaTransformação Extract stringtransformaçãofragmentsextraifrom a regular expression: extrair fragmentos de string usando uma expressão regular e criacriar uma nova coluna a partir dela, ou várias colunas se estiver usando grupos regex.
- Custom transform (Transformação personalizada): insira o código em um campo de entrada de texto para usar transformações personalizadas. A saída é uma coleção de `DynamicFrames`.

Usando uma fórmula de preparação de dados no AWS Glue Studio

O AWS Glue Studio permite que você use uma fórmula do AWS Glue DataBrew em um fluxo de trabalho visual. Isso permite que as fórmulas do AWS Glue DataBrew de um cliente sejam executadas em um trabalho do AWS Glue junto com outros nós do AWS Glue Studio.

No DataBrew, uma fórmula é um conjunto de etapas de transformação de dados. As fórmulas do DataBrew prescrevem como transformar dados que já foram lidos e não descrevem onde e como ler dados, nem como e onde gravá-los. Isso é configurado nos nós de origem e de destino no AWS Glue Studio. Para obter mais informações sobre fórmulas, consulte Creating and using AWS Glue DataBrew recipes.

O nó Fórmula de preparação de dados está disponível no painel Recursos. Você pode conectar o nó Fórmula de preparação de dados a outro nó no fluxo de trabalho visual, seja ele um nó de fonte de dados ou outro nó de transformação. Depois de escolher uma fórmula e uma versão do AWS Glue DataBrew, as etapas aplicadas na fórmula ficam visíveis na guia de propriedades do nó.

Pré-requisitos

- Você tem uma fórmula do AWS Glue DataBrew criada no AWS Glue DataBrew.
- Você tem as permissões de IAM de que você precisa, conforme descrito na seção abaixo.

Permissões do IAM para o AWS Glue DataBrew

Este tópico fornece informações para ajudar você a entender as ações e os recursos que você, um administrador de IAM pode usar em uma política do AWS Identity and Access Management (IAM) para a transformação Data Preparation Recipe.

Para obter mais informações sobre segurança no AWS Glue, consulte Access Management.

A tabela a seguir lista as permissões de que um usuário precisa para realizar operações específicas para usar a transformação Data Preparation Recipe.

Ações da transformação Data Preparaction Recipe

Ação	Descrição
`databrew:ListRecipes`	Concede permissão para recuperar fórmulas do AWS Glue DataBrew.
`databrew:ListRecipeVersions`	Concede permissão para recuperar versões de fórmulas do AWS Glue DataBrew.
`databrew:DescribeRecipe`	Concede permissão para recuperar descrição de fórmula do AWS Glue DataBrew.

O perfil que você está usando para acessar essa funcionalidade deve ter uma política que permita vários AWS Glue DataBrew Você pode fazer isso usando uma política do `AWSGlueConsoleFullAccess` que inclua as ações necessárias ou adicionar a seguinte política em linha ao seu perfil:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "databrew:ListRecipes",
                "databrew:ListRecipeVersions",
                "databrew:DescribeRecipe"
            ],
            "Resource": [
```

```
                "*"
            ]
        }
    ]
}
```

Para usar a transformação Data Preparation Recipe, você deve adicionar a ação `IAM:PassRole` à política de permissões.

Permissões adicionais necessárias

Ação	Descrição
`iam:PassRole`	Concede permissão ao IAM para permitir que o usuário passe os perfis aprovados.

Sem essas permissões, ocorre o seguinte erro:

```
"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"
```

Limitações

- Nem todas as fórmulas do AWS Glue DataBrew são compatíveis com o AWS Glue. Algumas fórmulas não poderão ser executadas no AWS Glue Studio.
  - Fórmulas com transformações UNION e JOIN não são suportadas, no entanto, o AWS Glue Studio já possui nós de transformação “Join” e “Union” que podem ser usados antes ou depois de um nó de Data Preparation Recipe.
- Os nós de Data Preparation Recipe são compatíveis com trabalhos a partir do AWS Glue versão 4.0. Essa versão será selecionada automaticamente depois que um nó de Data Preparation Recipe for adicionado ao trabalho.
- Nós Data Preparation Recipe exigem o Python. Isso é definido automaticamente quando o nó Data Preparation Recipe é adicionado ao trabalho.

- Ao usar a Visualização de Dados, você precisará reiniciar sua sessão de visualização de dados depois de adicionar um nó Data Preparation Recipe ao seu trabalho.

Como usar fórmulas do AWS Glue DataBrew no AWS Glue Studio

Para usar fórmulas do AWS Glue DataBrew no AWS Glue Studio, comece criando fórmulas no AWS Glue DataBrew. Se você já tiver fórmulas que deseja usar, pule esta etapa.

Para criar uma fórmula do AWS Glue DataBrew no AWS Glue DataBrew:

1. Crie uma fórmula no AWS Glue DataBrew. Para obter mais informações, consulte Getting started with AWS Glue DataBrew.
2. Salve a sua fórmula.
3. Publique a sua fórmula. Isso publicará sua fórmula como versão 1.0.

Para usar um nó de Data Preparation Recipe no AWS Glue Studio:

Você pode usar mais de um nó de Data Preparation Recipe em um trabalho visual de ETL. Para fazer isso, adicione um nó de Data Preparation Recipe seguindo as etapas abaixo e adicione outro nó de Data Preparation Recipe ao trabalho. Por exemplo, um fluxo de trabalho pode seguir esse padrão:

- Fonte de dados 1 > fórmula 1 > saída 1
- Fonte de dados 2 > fórmula 2 > saída 2
- saída 1, saída 2 > JOIN

1. Comece um trabalho do AWS Glue no AWS Glue Studio com uma origem de dados.
2. Adicione o nó Data Preparation Recipe à sua origem de dados.
3. Filtre a fórmula por nome digitando o nome da receita no campo de pesquisa.
4. Escolha a versão publicada. Apenas versões publicadas estão disponíveis.
5. Conclua a criação do trabalho adicionando outros nós de transformações conforme necessário e adicione nós de destino de dados para salvar a saída do trabalho.
6. Faça as alterações de configuração necessárias na guia Detalhes do trabalho, como nomear seu trabalho e ajustar a capacidade alocada conforme necessário, e salve o trabalho.

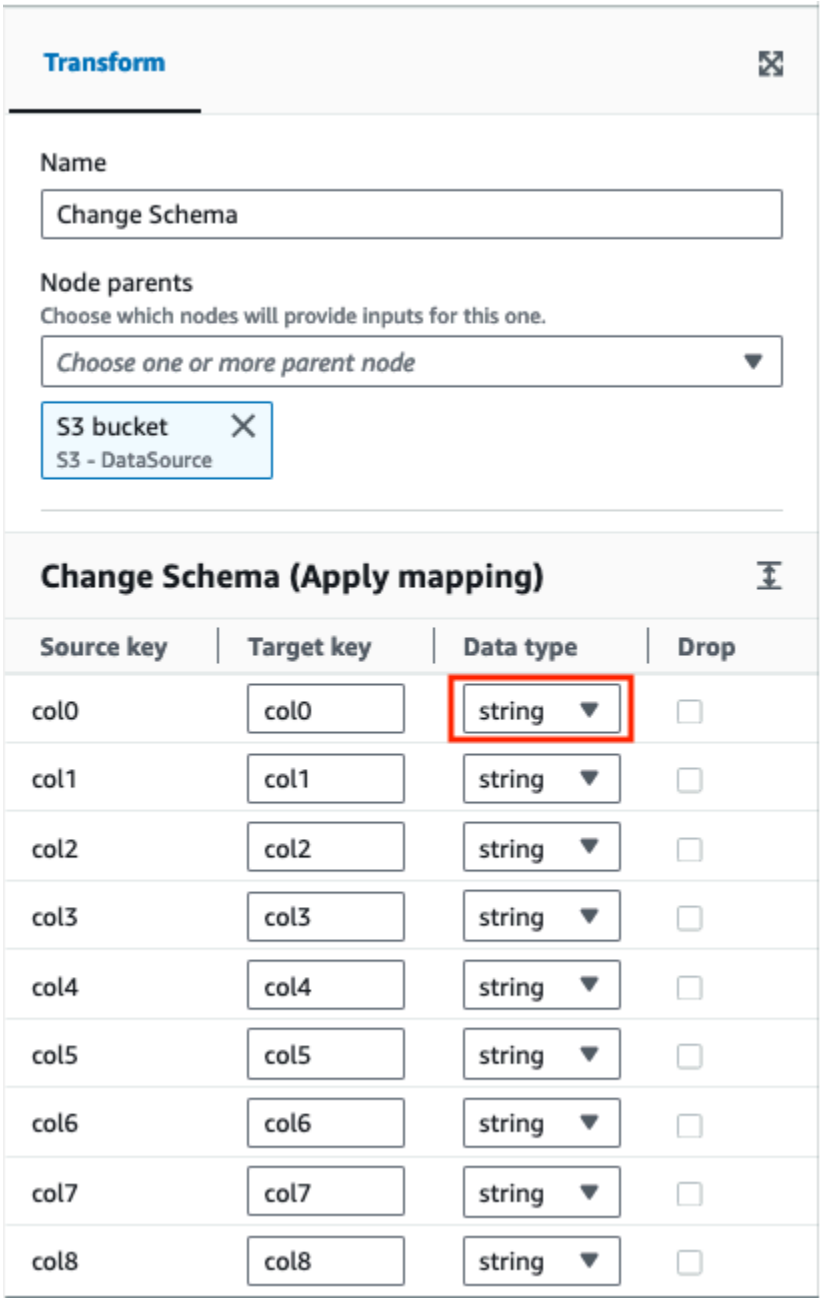
7. Execute o trabalho escolhendo Executar no menu suspenso Ações.

Para alterar o esquema se a fonte de dados for o Amazon S3 e o formato dos dados for CSV:

Se todas as colunas em um arquivo CSV forem inicialmente carregadas como tipo de dados de string no AWS Glue Studio, você precisará garantir que o tipo de dados da coluna seja compatível com o restante das etapas da fórmula do AWS Glue DataBrew.

As fórmulas do AWS Glue DataBrew prescrevem apenas como transformar dados que já foram lidos. Ela não descreve onde e como ler os dados.

1. Adicione um nó Change Schema antes do nó de fórmula de várias etapas.
2. Clique no nó Change Schema e altere o esquema para que seja o mesmo dos tipos de dados da coluna no AWS Glue DataBrew selecionando o novo tipo de dados em Transformar para colunas, conforme necessário.



Para alterar o esquema se a fonte de dados não tiver cabeçalho:

As fórmulas do AWS Glue DataBrew prescrevem apenas como transformar dados que já foram lidos. Ela não descreve onde e como ler os dados.

Ao carregar conjuntos de dados sem cabeçalho no AWS Glue Studio, os nomes dos cabeçalhos padrão são diferentes dos que são carregados no AWS Glue DataBrew.

1. No trabalho do ETL, adicione um nó Change Schema antes do nó Data Preparation Recipe.
2. Escolha o nó Change Schema e altere os nomes das colunas para os mesmos nomes usados na fórmula do AWS Glue DataBrew.

Usando Alterar esquema para remapear as chaves de propriedade de dados

Uma transformação Alterar esquema remapeia as chaves de propriedade de dados de origem como as chaves desejadas configuradas para os dados de destino. Em um nó de transformação Alterar esquema, você pode:

- Alterar o nome de várias chaves de propriedade de dados.
- Alterar o tipo de dados das chaves de propriedade de dados, se o novo tipo de dados for suportado e houver um caminho de transformação entre os dois tipos de dados.
- Escolher um subconjunto de chaves de propriedade de dados indicando quais chaves de propriedade de dados você deseja descartar.

Você também pode adicionar outros nós do Change Schema ao diagrama de tarefas conforme necessário — por exemplo, para modificar fontes de dados adicionais ou seguir uma transformação Join.

Usando o Change Schema com tipo de dados decimal

Ao usar a transformação Change Schema com tipo de dados decimal, a transformação Change Schema modifica a precisão para o valor padrão de (10,2). Para modificar isso e definir a precisão do seu caso de uso, você pode usar a transformação do SQL Query e converter as colunas com uma precisão específica.

Por exemplo, se você tiver uma coluna de entrada chamada "DecimalCol" do tipo Decimal e quiser remapeá-la para uma coluna de saída chamada "OutputDecimalCol" com uma precisão específica de (18,6), você poderia:

1. Adicione uma transformação subsequente do SQL Query após a transformação Change Schema.
2. Na transformação da consulta SQL, use uma consulta SQL para converter a coluna remapeada na precisão desejada. A consulta SQL ficaria assim:

```
SELECT col1, col2, CAST(DecimalCol AS DECIMAL(18,6)) AS OutputDecimalCol
```

```
FROM __THIS__
```

Na consulta SQL acima:

- `col1` e `col2` são outras colunas em seus dados que você deseja passar sem modificação.
- `DecimalCol` é o nome da coluna original dos dados de entrada.
- `CAST (DecimalCol AS DECIMAL (18,6)) `converte o` `em um tipo decimal com uma precisão de 18 dígitos e 6 casas decimais. DecimalCol
- `AS OutputDecimalCol `renomeia a coluna convertida para` `. OutputDecimalCol

Usando a transformação do SQL Query, você pode substituir a precisão padrão definida pela transformação Change Schema e converter explicitamente as colunas decimais com a precisão desejada. Essa abordagem permite que você aproveite a transformação Change Schema para renomear e reestruturar seus dados e, ao mesmo tempo, lidar com os requisitos de precisão das colunas decimais por meio da transformação subsequente do SQL Query.

Adicionar uma transformação do Change Schema ao seu trabalho

> **Note**
>
> A transformação Alterar esquema não diferencia maiúsculas de minúsculas.

Para adicionar um nó de transformação Alterar esquema ao diagrama de trabalho

1. (Opcional) abra o painel Recurso e escolha Alterar esquema para adicionar uma nova transformação ao diagrama de trabalho, se necessário.
2. Na guia Propriedades do nó, insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Escolha a guia Transformar no painel de detalhes do nó.
4. Modifique o esquema de entrada:

   - Para renomear uma chave de propriedade de dados, insira o novo nome da chave no campo Target key (Chave de destino).
   - Para alterar o tipo de dados de uma chave de propriedade de dados, escolha o novo tipo de dados para a chave na lista Data type (Tipo de dados).

- Para remover uma chave de propriedade de dados do esquema de destino, marque a caixa de seleção Drop (Descartar) da chave correspondente.

5. (Opcional) depois de configurar as propriedades do nó de transformação, você pode visualizar o esquema modificado dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.
6. (Opcional) depois de configurar as propriedades do nó e da transformação, você pode previsualizar o conjunto de dados modificado escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Usando Drop Duplicates

A transformação Drop Duplicates remove linhas da sua fonte de dados, oferecendo duas opções. Você pode optar por remover a linha duplicada que é completamente igual ou escolher os campos a serem correspondentes e remover somente as linhas com base nos campos escolhidos.

Por exemplo, nesse conjunto de dados, você tem linhas duplicadas em que todos os valores em algumas linhas são exatamente iguais aos de outra linha e alguns dos valores nas linhas são iguais ou diferentes.

Linha	Nome	E-mail	Idade	State	Observação
1	Joy	joy@gmail	33	NY	
2	Tim	tim@gmail	45	OH	
3	Rose	rose@gmail	23	NJ	
4	Tim	tim@gmail	42	OH	
5	Rose	rose@gmail	23	NJ	

Linha	Nome	E-mail	Idade	State	Observação
6	Tim	tim@gmail	42	OH	esta é uma linha duplicada e corresponde completamente a todos os valores da linha #4
7	Rose	rose@gmail	23	NJ	Esta é uma linha duplicada e corresponde completamente a todos os valores da linha #5

Se você optar por combinar linhas inteiras, as linhas 6 e 7 serão removidas do conjunto de dados. O conjunto de dados agora é:

Linha	Nome	E-mail	Idade	State
1	Joy	joy@gmail	33	NY
2	Tim	tim@gmail	45	OH
3	Rose	rose@gmail	23	NJ
4	Tim	tim@gmail	42	OH
5	Rose	rose@gmail	23	NJ

Se você optar por especificar as chaves, poderá optar por remover as linhas que coincidem em 'nome' e 'e-mail'. Isso oferece um controle mais preciso do que é uma “linha duplicada” para seu conjunto de dados. Ao especificar 'nome' e 'e-mail', o conjunto de dados agora é:

Linha	Nome	E-mail	Idade	State
1	Joy	joy@gmail	33	NY
2	Tim	tim@gmail	45	OH
3	Rose	rose@gmail	23	NJ

Tenha em mente que:

- Para que as linhas sejam reconhecidas como duplicadas, os valores diferenciam maiúsculas de minúsculas. Todos os valores nas linhas precisam ter a mesma letra maiúscula e minúscula - isso se aplica a qualquer opção escolhida (Combinar linhas inteiras ou Especificar chaves).
- Todos os valores são lidos como cadeias de caracteres.
- A transformação Drop Duplicates utiliza o comando dropDuplicates do Spark.
- Ao usar a transformação Drop Duplicates, a primeira linha é mantida e as outras linhas são descartadas.
- A transformação Drop Duplicates não altera o esquema do dataframe. Se você optar por especificar as chaves, todos os campos serão mantidos no dataframe resultante.

Usar SelectFields (Selecionar campos) para remover a maioria das chaves de propriedade de dados

Você pode criar um subconjunto de chaves de propriedade de dados a partir do conjunto de dados usando a transformação SelectFields (Selecionar campos). Você indica quais chaves de propriedade de dados deseja manter e o restante é removido do conjunto de dados.

> **Note**
>
> A transformação SelectFields (Selecionar campos) diferencia maiúsculas de minúsculas. Use ApplyMapping (Aplicar mapeamentos) se você precisar de uma maneira de selecionar campos que não diferencie maiúsculas de minúsculas.

Para adicionar um nó de transformação SelectFields ao diagrama de tarefas

1. (Opcional) abra o painel Recurso e escolha SelectFields) para adicionar uma nova transformação ao diagrama de trabalho, se necessário.
2. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Escolha a guia Transform (Transformação) no painel de detalhes do nó.
4. No cabeçalho SelectFields (Selecionar campos), escolha as chaves de propriedade de dados no conjunto de dados que você deseja manter. Quaisquer chaves de propriedade de dados não selecionadas são descartadas do conjunto de dados.

   Você também pode marcar a caixa de seleção ao lado do cabeçalho de coluna Field (Campo) para escolher automaticamente todas as chaves de propriedade de dados do conjunto de dados. Em seguida, você pode desmarcar chaves de propriedade de dados individuais para removê-las do conjunto de dados.
5. (Opcional) depois de configurar as propriedades do nó de transformação, você pode visualizar o esquema modificado dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.
6. (Opcional) depois de configurar as propriedades do nó e da transformação, você pode previsualizar o conjunto de dados modificado escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Usar DropFields (Descartar campos) para manter a maioria das chaves de propriedade de dados

Você pode criar um subconjunto de chaves de propriedade de dados a partir do conjunto de dados usando a transformação DropFields (Descartar campos). Você indica quais chaves de propriedade de dados deseja remover e o restante é mantido no conjunto de dados.

> **Note**
>
> A transformação DropFields (Descartar campos) diferencia maiúsculas de minúsculas. Use ApplyMapping se você precisar de uma maneira de selecionar campos que não diferencie maiúsculas de minúsculas.

Para adicionar um nó de transformação DropFields (Descartar campos) ao diagrama de trabalho

1. (Opcional) abra o painel Recurso e escolha DropFields) para adicionar uma nova transformação ao diagrama de trabalho, se necessário.
2. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Escolha a guia Transform (Transformação) no painel de detalhes do nó.
4. No cabeçalho DropFields (Descartar campos), escolha as chaves de propriedade de dados a serem descartadas da origem dos dados.

   Você também pode marcar a caixa de seleção ao lado do cabeçalho de coluna Field (Campo) para escolher automaticamente todas as chaves de propriedade de dados do conjunto de dados. Em seguida, você pode cancelar a seleção de chaves de propriedade de dados individuais para que elas sejam mantidas no conjunto de dados.
5. (Opcional) depois de configurar as propriedades do nó de transformação, você pode visualizar o esquema modificado dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.
6. (Opcional) depois de configurar as propriedades do nó e da transformação, você pode previsualizar o conjunto de dados modificado escolhendo a guia Data preview (Previsualização

de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Renomear um campo no conjunto de dados

Você pode usar a transformação RenameField (Renomear campo) para alterar o nome de uma chave de propriedade individual no conjunto de dados.

> **Note**
>
> A transformação RenameField (Renomear campo) diferencia maiúsculas de minúsculas. Use ApplyMapping (Aplicar mapeamento), se você precisar de uma transformação sem distinção entre maiúsculas e minúsculas.

> **Tip**
>
> Se você usar a transformação ApplyMapping, poderá renomear várias chaves de propriedade de dados no conjunto de dados com uma única transformação.

Para adicionar um nó de transformação RenameField (Renomear campo) ao diagrama de trabalho

1. (Opcional) abra o painel Recurso e escolha RenameField para adicionar uma nova transformação ao diagrama de trabalho, se necessário.
2. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Escolha a guia Transform (Transformação).
4. No cabeçalho Data field (Campo de dados), escolha uma chave de propriedade nos dados de origem e, em seguida, insira um novo nome no campo New name field (Novo nome de campo).
5. (Opcional) depois de configurar as propriedades do nó de transformação, você pode visualizar o esquema modificado dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar

os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.

6. (Opcional) depois de configurar as propriedades do nó e da transformação, você pode previsualizar o conjunto de dados modificado escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Usar Spigot (Torneira) para fazer uma amostra de seu conjunto de dados

Para testar as transformações executadas pelo trabalho, convém obter uma amostra dos dados para verificar se a transformação funciona como pretendido. A transformação Spigot (Torneira) grava um subconjunto de registros do conjunto de dados em um arquivo JSON, em um bucket do Amazon S3. O método de amostragem de dados pode ser um número especificado de registros desde o início do arquivo ou um fator de probabilidade usado para selecionar registros.

Para adicionar um nó de transformação Spigot (Torneira) ao diagrama de trabalho

1. (Opcional) abra o painel Recurso e escolha Spigot para adicionar uma nova transformação ao diagrama de trabalho, se necessário.
2. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Escolha a guia Transform (Transformação) no painel de detalhes do nó.
4. Insira um caminho do Amazon S3 ou escolha Browse S3 (Procurar no S3) para escolher um local no Amazon S3. Esse é o local onde o trabalho grava o arquivo JSON que contém a amostra de dados.
5. Insira informações para o método de amostragem. É possível especificar um valor em Number of records (Número de registros) para gravar a partir do início do conjunto de dados e um Probability threshold (Limite de probabilidade, inserido como um valor decimal com um valor máximo de 1) de escolher qualquer registro dado.

   Por exemplo, para gravar os primeiros 50 registros do conjunto de dados, você definiria Number of records (Número de registros) como 50 e Probability threshold (Limite de probabilidade) como 1 (100%).

Unir conjuntos de dados

A transformação Join (Unir) permite combinar dois conjuntos de dados em um. Você especifica os nomes das chaves no esquema de cada conjunto de dados a ser comparado. O `DynamicFrame` de saída contém linhas em que as chaves atendem à condição de união. As linhas em cada conjunto de dados que atendem à condição de união são combinadas em uma única linha no `DynamicFrame` de saída que contém todas as colunas encontradas em qualquer um dos conjuntos de dados.

Para adicionar um nó de transformação Join (Unir) ao diagrama de trabalho

1. Se houver apenas uma fonte de dados disponível, você deverá adicionar um novo nó de origem dos dados ao diagrama de trabalho.
2. Escolha um dos nós de origem para a união. Abra o painel Recurso e escolha Unir para adicionar uma nova transformação ao diagrama do trabalho.
3. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho.
4. Na guia Node properties (Propriedades do nó), no cabeçalho Node parents (Nós pais), adicione um nó pai para que haja dois conjuntos de dados fornecendo entradas para a união. O pai pode ser um nó de origem dos dados ou um nó de transformação.

   > **Note**
   >
   > Uma união pode ter apenas dois nós pais.

5. Escolha a guia Transform (Transformação).

   Se você vir uma mensagem indicando que existem nomes de chaves conflitantes, você pode:

   - Escolher Resolve it (Resolver) para adicionar automaticamente um nó de transformação ApplyMapping (Aplicar mapeamento) em seu diagrama de trabalho. O nó ApplyMapping (Aplicar mapeamento) adiciona um prefixo a todas as chaves no conjunto de dados que tenham o mesmo nome de uma chave do outro conjunto de dados. Por exemplo, se você usar o valor padrão **`right`**, todas as chaves no conjunto de dados direito que tenham o mesmo nome de uma chave do conjunto de dados esquerdo serão renomeadas para `(right)`*`key name`*.
   - Adicionar manualmente um nó de transformação anteriormente no diagrama de trabalho para remover ou renomear as chaves conflitantes.

6. Escolha o tipo de junção na lista Join type (Tipo de união).

- Inner join (União interna): retorna uma linha com colunas de ambos os conjuntos de dados para cada correspondência com base na condição de união. As linhas que não satisfazem a condição de união não são retornadas.
- Left join (União à esquerda): todas as linhas do conjunto de dados esquerdo e somente as linhas do conjunto de dados direito que satisfazem a condição de união.
- Right join (União à direita): todas as linhas do conjunto de dados direito e somente as linhas do conjunto de dados esquerdo que satisfazem a condição de união.
- Outer join (União externa): todas as linhas de ambos os conjuntos de dados.
- Left semi join (semi à esquerda): todas as linhas do conjunto de dados esquerdo que têm uma correspondência no conjunto de dados direito com base na condição de união.
- Left anti join (anti à esquerda): todas as linhas do conjunto de dados esquerdo que não têm uma correspondência no conjunto de dados direito com base na condição de união.

7. Na guia Transform (Transformação), no cabeçalho Join conditions (Condições de união), escolha Add condition (Adicionar condição). Escolha uma chave de propriedade de cada conjunto de dados para comparar. As chaves de propriedade no lado esquerdo do operador de comparação são referidas como o conjunto de dados esquerdo e as chaves de propriedade à direita são referidas como o conjunto de dados direito.

   Para condições de união mais complexas, você pode adicionar chaves correspondentes adicionais escolhendo Add condition (Adicionar condição) mais de uma vez. Se você adicionar uma condição acidentalmente, pode escolher o ícone de exclusão ( ) para removê-la.

8. (Opcional) depois de configurar as propriedades do nó de transformação, você pode visualizar o esquema modificado dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.
9. (Opcional) depois de configurar as propriedades do nó e da transformação, você pode previsualizar o conjunto de dados modificado escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM

para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Para obter um exemplo de esquema de saída de união, considere uma união entre dois conjuntos de dados com as seguintes chaves de propriedade:

```
Left: {id, dept, hire_date, salary, employment_status}
Right: {id, first_name, last_name, hire_date, title}
```

A união está configurada para corresponder nas chaves `id` e `hire_date` usando o operador de comparação =.

Como ambos os conjuntos de dados contêm as chaves `id` e `hire_date`, você escolhe Resolve it (Resolver) para adicionar automaticamente o prefixo **`right`** às chaves no conjunto de dados direito.

As chaves no esquema de saída seriam:

```
{id, dept, hire_date, salary, employment_status,
(right)id, first_name, last_name, (right)hire_date, title}
```

Usando Union para combinar linhas

Você usa o nó de transformação Union quando deseja combinar linhas de mais de uma fonte de dados que tenham o mesmo esquema.

Existem dois tipos de transformações Union:

1. ALL - ao aplicar ALL, a união resultante não remove linhas duplicadas.
2. DISTINCT: ao aplicar DISTINCT, a união resultante remove as linhas duplicadas.

Unions versus Joins

Você usa Union para combinar linhas. Você usa Join para combinar colunas.

Usando a transformação Union na tela Visual ETL

1. Adicione mais de uma fonte de dados para realizar uma transformação de união. Para adicionar uma fonte de dados, abra o painel Recurso e escolha a fonte de dados na guia Fontes. Antes de usar a transformação Union, você deve garantir que todas as fontes de dados envolvidas na união tenham o mesmo esquema e estrutura.

2. Quando você tem pelo menos duas fontes de dados que deseja combinar usando a transformação Union, crie a transformação Union adicionando-a à tela. Abra o painel Recurso na tela e pesquise “Union”. Você também pode escolher a guia Transformações no painel Recursos e rolar para baixo até encontrar a transformação Union e, em seguida, escolher Union.
3. Selecione o nó Union na tela de trabalho. Na janela Propriedades do nó, escolha os nós principais para se conectar à transformação Union.
4. O AWS Glue verifica a compatibilidade para garantir que a transformação Union possa ser aplicada a todas as fontes de dados. Se o esquema das fontes de dados for o mesmo, a operação será permitida. Se as fontes de dados não tiverem o mesmo esquema, uma mensagem de erro de inválido será exibida: “Os esquemas de entrada dessa união não são os mesmos. Considere usar o ApplyMapping para combinar os esquemas.” Para corrigir isso, escolha usar ApplyMapping.
5. Escolha o tipo de união.

    1. All: por padrão, o tipo de Union All é selecionado; isso resultará em linhas duplicadas, se houver alguma na combinação de dados.
    2. Distinct: escolha Distinct se quiser que as linhas duplicadas sejam removidas da combinação de dados resultante.

Usar SplitFields (Dividir campos) para dividir um conjunto de dados em dois

A transformação SplitFields (Dividir campos) permite que você escolha algumas das chaves de propriedade de dados no conjunto de dados de entrada, colocando-as em um conjunto de dados e as chaves não selecionadas em um conjunto de dados separado. A saída dessa transformação é uma coleção de `DynamicFrames`.

> **Note**
>
> Você deve usar uma transformação SelectFromCollection (Selecionar da coleção) para converter a coleção de `DynamicFrames` em um único `DynamicFrame` antes que você possa enviar a saída para um local de destino.

A transformação SplitFields (Dividir campos) diferencia maiúsculas de minúsculas. Adicione uma transformação ApplyMapping (Aplicar mapeamento) como um nó pai, se você precisar de nomes de chave de propriedade sem distinção entre maiúsculas e minúsculas.

Para adicionar um nó de transformação SplitFields (Dividir campos) ao diagrama de trabalho

1. (Opcional) abra o painel Recurso e escolha Dividir campos para adicionar uma nova transformação ao diagrama de trabalho, se necessário.
2. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Escolha a guia Transform (Transformação).
4. Escolha quais chaves de propriedade você deseja colocar no primeiro conjunto de dados. As chaves que você não escolher serão colocadas no segundo conjunto de dados.
5. (Opcional) depois de configurar as propriedades do nó de transformação, você pode visualizar o esquema modificado dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.
6. (Opcional) depois de configurar as propriedades do nó e da transformação, você pode previsualizar o conjunto de dados modificado escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.
7. Configure um nó de transformação SelectFromCollection (Selecionar da coleção) para processar os conjuntos de dados resultantes.

Visão geral da transformação SelectFromCollection

Determinadas transformações têm vários conjuntos de dados como saída, em vez de um único conjunto de dados, por exemplo, SplitFields (Dividir campos). A transformação SelectFromCollection (Selecionar da coleção) seleciona um conjunto de dados (`DynamicFrame`) em uma coleção de conjuntos de dados (uma matriz de `DynamicFrames`). A saída da transformação é o `DynamicFrame` selecionado.

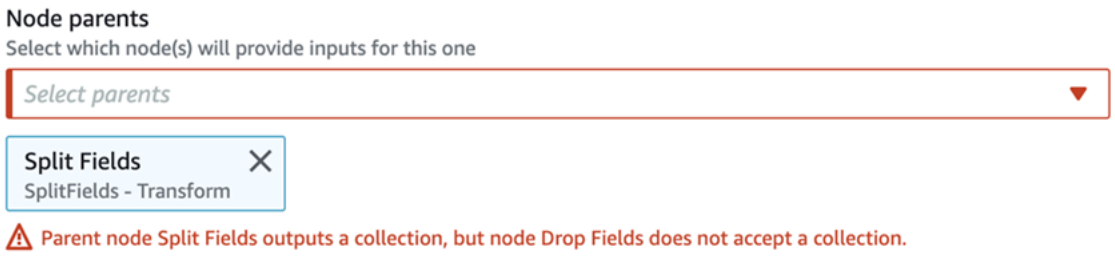
Você deve usá-la após usar uma transformação que cria uma coleção de `DynamicFrames`, tais como:

- Transformações de código personalizado
- SplitFields

Se você não adicionar um nó de transformação SelectFromCollection (Selecionar da coleção) em seu diagrama de trabalho após qualquer uma dessas transformações, você receberá um erro em seu trabalho.

O nó pai para essa transformação deve ser um nó que retorna uma coleção de `DynamicFrames`. Se você escolher um pai para esse nó de transformação que retorne um único `DynamicFrame`, como uma transformação Join (Unir), seu trabalho retornará um erro.

Da mesma forma, se você usar um nó SelectFromCollection (Selecionar da coleção) em seu diagrama de trabalho como pai para uma transformação que espera um único `DynamicFrame` como entrada, o trabalho retornará um erro.



Usar SelectFromCollection (Selecionar da coleção) para escolher qual conjunto de dados manter

Use a transformação SelectFromCollection (Selecionar da coleção) para converter uma coleção de `DynamicFrames` em um único `DynamicFrame`.

Para adicionar um nó de transformação SelectFromCollection (Selecionar da coleção) ao diagrama de tarefas

1. (Opcional) Abra o painel Recurso e escolha SelectFromCollection) para adicionar uma nova transformação ao diagrama de trabalho, se necessário.
2. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Escolha a guia Transform (Transformação).

4. No cabeçalho Frame index (Índice de quadros), escolha o número do índice da matriz que corresponde ao `DynamicFrame` que deseja selecionar na coleção de `DynamicFrames`.

   Por exemplo, se o nó pai para essa transformação for uma transformação SplitFields (Dividir campos), na guia Output schema (Esquema de saída) desse nó, você pode ver o esquema de cada `DynamicFrame`. Se quiser manter o `DynamicFrame` associado ao esquema de Output 2, você pode selecionar **1** para o valor de Frame index (Índice de quadros), que é o segundo valor na lista.

   Somente o `DynamicFrame` que você escolher será incluído na saída.

5. (Opcional) depois de configurar as propriedades do nó de transformação, você pode visualizar o esquema modificado dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.
6. (Opcional) depois de configurar as propriedades do nó e da transformação, você pode previsualizar o conjunto de dados modificado escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Localizar e preencher valores ausentes em um conjunto de dados

Você pode usar a transformação FillMissingValues (Preencher valores ausentes) para localizar registros no conjunto de dados que têm valores ausentes e adicionar um novo campo com um valor determinado por imputação. O conjunto de dados de entrada é usado para treinar o modelo de machine learning (ML) que determina qual deve ser o valor ausente. Se você usar conjuntos de dados incrementais, cada conjunto incremental será usado como dados de treinamento para o modelo de ML, portanto, os resultados podem não ser tão precisos.

Para adicionar um nó de transformação FillMissingValues (Preencher valores ausentes) ao diagrama de trabalho

1. (Opcional) abra o painel Recurso e escolha FillMissingValues) para adicionar uma nova transformação ao diagrama de trabalho, se necessário.

2. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Escolha a guia Transform (Transformação).
4. Em Data field (Campo de dados), escolha o nome da coluna ou do campo nos dados de origem que você deseja analisar quanto a valores ausentes.
5. (Opcional) no campo New field name (Nome do novo campo), insira um nome para o campo adicionado a cada registro que manterá o valor estimado de substituição para o campo analisado. Se o campo analisado não tiver um valor ausente, o valor no campo analisado será copiado para o novo campo.

   Se você não especificar um nome para o novo campo, o nome padrão será o nome da coluna analisada seguido de `_filled`. Por exemplo, se você inserir **`Age`** em Data field (Campo de dados) e não especificar um valor para New field name (Nome do novo campo), um novo campo chamado **`Age_filled`** será adicionado a cada registro.
6. (Opcional) depois de configurar as propriedades do nó de transformação, você pode visualizar o esquema modificado dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.
7. (Opcional) depois de configurar as propriedades do nó e da transformação, você pode previsualizar o conjunto de dados modificado escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Filtrar chaves dentro de um conjunto de dados

Use a transformação Filter (Filtro) para criar um novo conjunto de dados filtrando registros do conjunto de dados de entrada com base em uma expressão regular. As linhas que não satisfaçam a condição do filtro são removidas da saída.

- Para tipos de dados de string, você pode filtrar linhas em que o valor da chave corresponde a uma string especificada.

- Para tipos de dados numéricos, você pode filtrar linhas comparando o valor da chave com um valor especificado usando os operadores de comparação <, >, =, !=, <= e >=.

Se você especificar várias condições de filtro, os resultados serão combinados usando um operador AND por padrão, mas você pode escolher OR em vez disso.

A transformação Filter (Filtro) diferencia maiúsculas de minúsculas. Adicione uma transformação ApplyMapping (Aplicar mapeamento) como um nó pai, se você precisar de nomes de chave de propriedade sem distinção entre maiúsculas e minúsculas.

Para adicionar um nó de transformação Filter (Filtro) ao diagrama de trabalho

1. (Opcional) Abra o painel Recurso e escolha Filtro para adicionar uma nova transformação ao diagrama de trabalho, se necessário.
2. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Escolha a guia Transform (Transformação).
4. Escolha Global AND ou Global OR. Isso determina como várias condições de filtro são combinadas. Todas as condições são combinadas usando os operadores AND ou OR. Se você tiver apenas uma única condição de filtro, pode escolher qualquer uma.
5. Escolha o botão Add condition (Adicionar condição) na seção Filter condition (Condição de filtro) para adicionar uma condição de filtro.

   No campo Key (Chave), escolha um nome de chave de propriedade do conjunto de dados. No campo Operation (Operação), escolha o operador de comparação. No campo Value (Valor), insira o valor de comparação. Estes são alguns exemplos de configurações de filtro:

   - `year >= 2018`
   - `State matches 'CA*'`

   Ao filtrar valores de string, certifique-se de que o valor de comparação usa um formato de expressão regular que corresponda à linguagem de script selecionada nas propriedades do trabalho (Python ou Scala).
6. Adicione mais condições de filtro, conforme necessário.

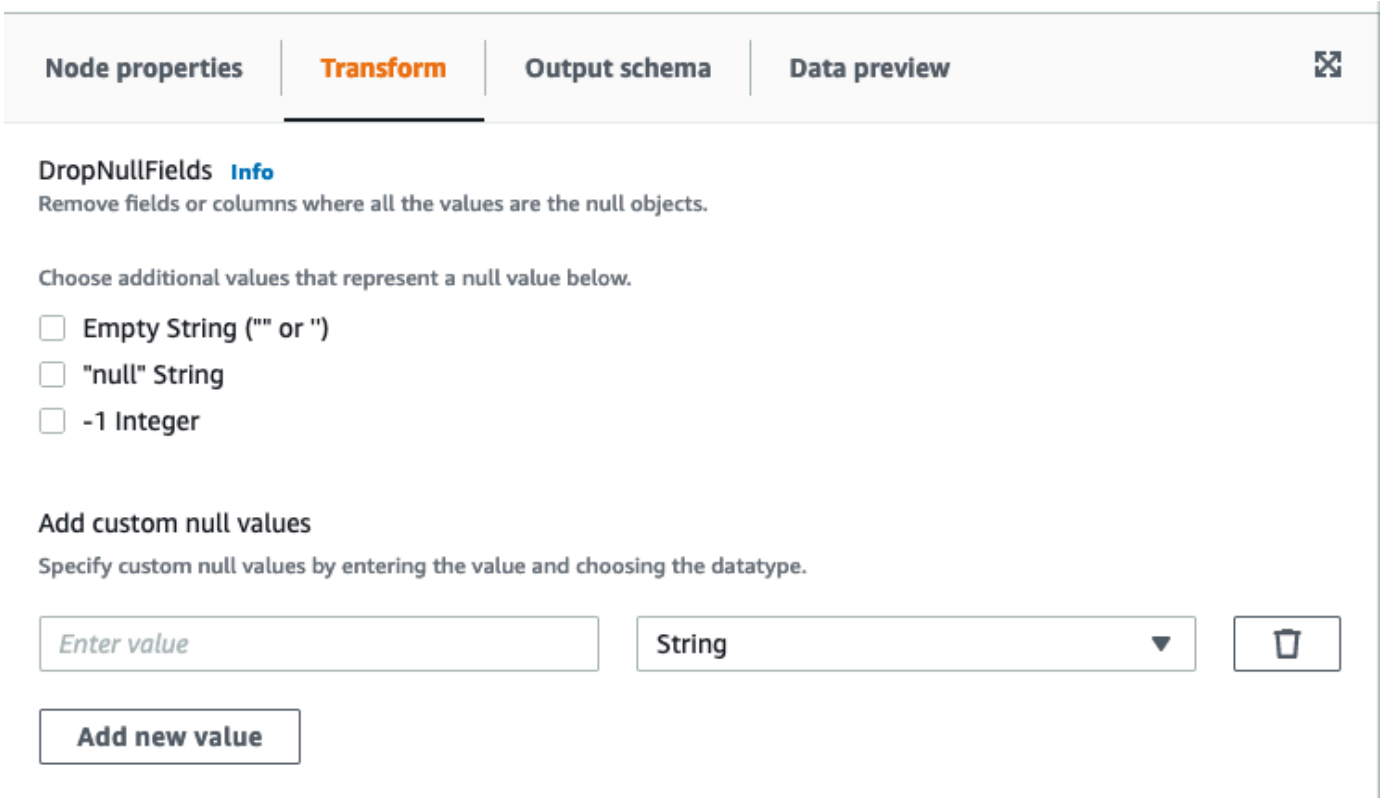
7. (Opcional) depois de configurar as propriedades do nó de transformação, você pode visualizar o esquema modificado dos dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.
8. (Opcional) depois de configurar as propriedades do nó e da transformação, você pode previsualizar o conjunto de dados modificado escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Usar DropNullFields para remover campos com valores nulos

Use a transformação DropNullFields para remover campos do conjunto de dados se todos os valores no campo forem "null". Por padrão, o AWS Glue Studio reconhecerá objetos nulos, mas alguns valores como strings vazias, strings "null", inteiros -1 ou outros espaços reservados, como zeros, não são reconhecidos automaticamente como nulos.

Para usar DropNullFields

1. Adicione um nó DropNullFields ao diagrama do trabalho.
2. Na guia Node properties (Propriedades do nó), escolha valores adicionais que representem um valor nulo. Você pode optar por selecionar nenhum ou todos os valores:



- String vazia ("" ou ''): os campos que contiverem strings vazias serão removidos
- "String null": os campos que contiverem a string com a palavra "null" serão removidos
- Inteiro -1: os campos que contiverem um número inteiro -1 (menos um) serão removidos

3. Se necessário, você também poderá especificar valores nulos personalizados. Esses são valores nulos que podem ser exclusivos do seu conjunto de dados. Para adicionar um valor nulo personalizado, escolha Add new value (Adicionar novo valor).
4. Insira o valor nulo personalizado. Por exemplo, ele pode zero ou qualquer valor que esteja sendo usado para representar um nulo no conjunto de dados.
5. Escolha o tipo de dados no campo suspenso. Os tipos de dados podem ser String ou Integer.

> **Note**
>
> Os valores nulos personalizados e seus tipos de dados devem corresponder exatamente para que os campos sejam reconhecidos como valores nulos e os campos removidos. Correspondências parciais em que apenas o valor nulo personalizado corresponde, mas o tipo de dados não resultará na remoção dos campos.

Usar uma consulta SQL para transformar dados

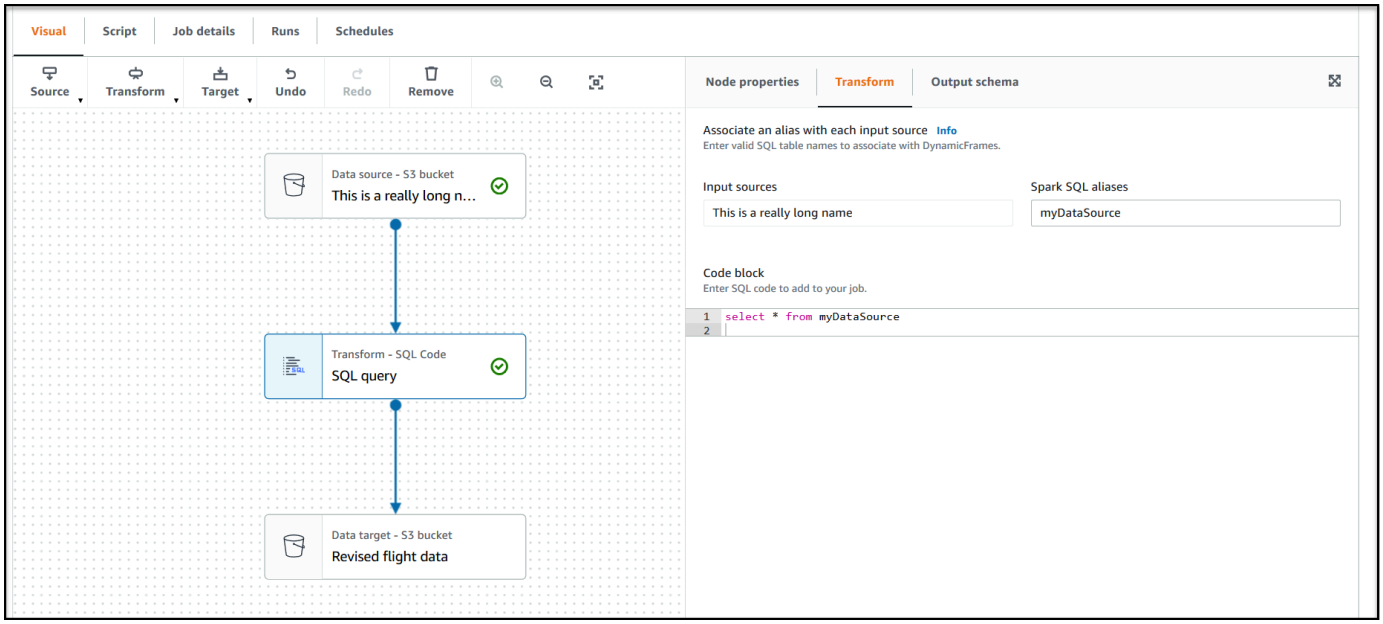
Você pode usar uma transformação SQL para escrever sua própria transformação na forma de uma consulta SQL.

Um nó de transformação SQL pode ter vários conjuntos de dados como entradas, mas produz apenas um único conjunto de dados como saída. Ele contém um campo de texto, onde você insere a consulta do Apache SparkSQL. Você pode atribuir aliases a cada conjunto de dados usado como entrada, para ajudar a simplificar a consulta SQL. Para obter mais informações sobre a sintaxe SQL, consulte a documentação do Spark SQL.

> **Note**
>
> Se você usar uma transformação do Spark SQL com uma origem dos dados localizada em uma VPC, adicione um endpoint da VPC do AWS Glue à VPC que contém a origem dos dados. Para obter mais informações sobre como configurar endpoints de desenvolvimento, consulte Adicionar um endpoint de desenvolvimento, Configurar seu ambiente para endpoints de desenvolvimento e Acessar o endpoint de desenvolvimento no Guia do desenvolvedor do AWS Glue.

Para usar um nó de transformação SQL no diagrama de trabalho

1. (Opcional) adicione um nó de transformação ao diagrama de trabalho, se necessário. Escolha Spark SQL para o tipo de nó.
2. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho. Se um nó pai ainda não estiver selecionado, ou se você quiser várias entradas para a transformação SQL, escolha um nó na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação. Adicione nós pais conforme necessário.
3. Escolha a guia Transform (Transformação) no painel de detalhes do nó.
4. Os conjuntos de dados de origem para a consulta SQL são identificados pelos nomes especificados no campo Name (Nome) de cada nó. Se você não quiser usar esses nomes, ou se eles não forem adequados para uma consulta SQL, você pode associar um nome a cada conjunto de dados. O console fornece aliases padrão, como `MyDataSource`.



Por exemplo, se um nó pai do nó de transformação SQL for nomeado como `Rename Org PK field`, você poderá associar o nome `org_table` com este conjunto de dados. Esse alias pode ser usado na consulta SQL no lugar do nome do nó.

5. No campo de entrada de texto, no cabeçalho Code block (Bloco de código), cole ou insira a consulta SQL. O campo de texto exibe o realce da sintaxe SQL e sugestões de palavra-chave.
6. Com o nó de transformação SQL selecionado, escolha a guia Output schema (Esquema de saída) e, em seguida, escolha Edit (Editar). Forneça as colunas e os tipos de dados que descrevem os campos de saída da consulta SQL.

   Especifique o esquema usando as seguintes ações na seção Output schema (Esquema de saída) da página:

   - Para renomear uma coluna, coloque o cursor na caixa de texto Key (Chave) da coluna (também conhecida como field (campo) ou property key [chave de propriedade]) e insira o novo nome.
   - Para alterar o tipo de dados de uma coluna, selecione o novo tipo de dados para a coluna na lista suspensa.
   - Para adicionar uma nova coluna de nível superior ao esquema, escolha o botão Overflow (excedente,

)

     e, em seguida, escolha Add root key (Adicionar chave raiz). Novas colunas são adicionadas na parte superior do esquema.

- Para remover uma coluna do esquema, escolha o ícone de exclusão
  ( )
  à extrema direita do nome da chave.

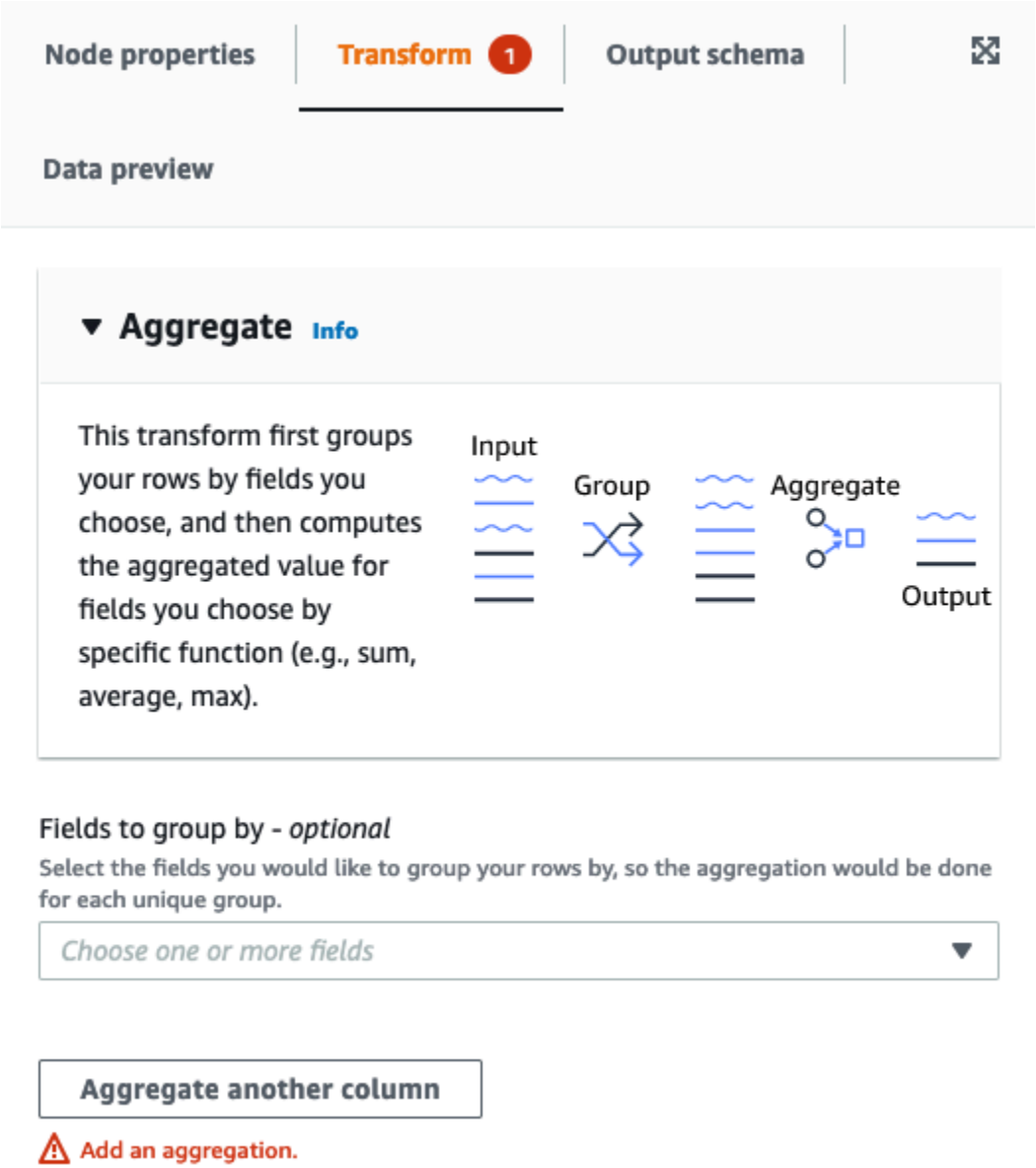
7. Quando terminar de especificar o esquema de saída, escolha Apply (Aplicar) para salvar suas alterações e saia do editor de esquema. Se não quiser salvar as alterações, escolha Cancel (Cancelar) para editar o editor de esquemas.
8. (Opcional) depois de configurar as propriedades do nó e da transformação, você pode previsualizar o conjunto de dados modificado escolhendo a guia Data preview (Previsualização de dados) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Há um custo associado ao uso desse recurso e o a cobrança começa assim que você fornece uma função do IAM.

Usar Aggregate para fazer cálculos resumidos em campos selecionados

Para usar a transformação Aggregate

1. Adicione o nó Aggregate ao diagrama do trabalho.
2. Na guia Node properties (Propriedades de nó), escolha campos para agrupar selecionando o campo suspenso (opcional). Você pode selecionar mais de um campo por vez ou procurar um nome de campo digitando na barra de pesquisa.

   Quando os campos são selecionados, o nome e o tipo de dados são mostrados. Para remover um campo, escolha "X" no campo.



3. Selecione Aggregate another column (Agregar outra coluna). É necessário selecionar pelo menos um campo.

4. Escolha um campo na lista suspensa Field to aggregate (Campo a ser agregado).

5. Escolha a função de agregação a ser aplicada ao campo escolhido:

   - avg: calcula a média
   - CountDistinct: calcula o número de valores não nulos exclusivos
   - count: calcula o número de valores não nulos
   - first: retorna o primeiro valor que satisfaz os critérios "agrupar por"
   - last: retorna o último valor que satisfaz os critérios "agrupar por"
   - kurtosis: calcula a nitidez do pico de uma curva de distribuição de frequência
   - max: retorna o maior valor que satisfaz os critérios "agrupar por"
   - min: retorna o menor valor que satisfaz os critérios "agrupar por"
   - skewness: medida da assimetria da distribuição de probabilidade de uma distribuição normal
   - stddev_pop: calcula o desvio padrão da população e retorna a raiz quadrada da variância da população
   - sum: a soma de todos os valores no grupo
   - sumDistinct: a soma dos valores distintos no grupo
   - var_samp: a variância da amostra do grupo (ignora nulos)
   - var_pop: a variância da população do grupo (ignora nulos)

Nivelar structs aninhados

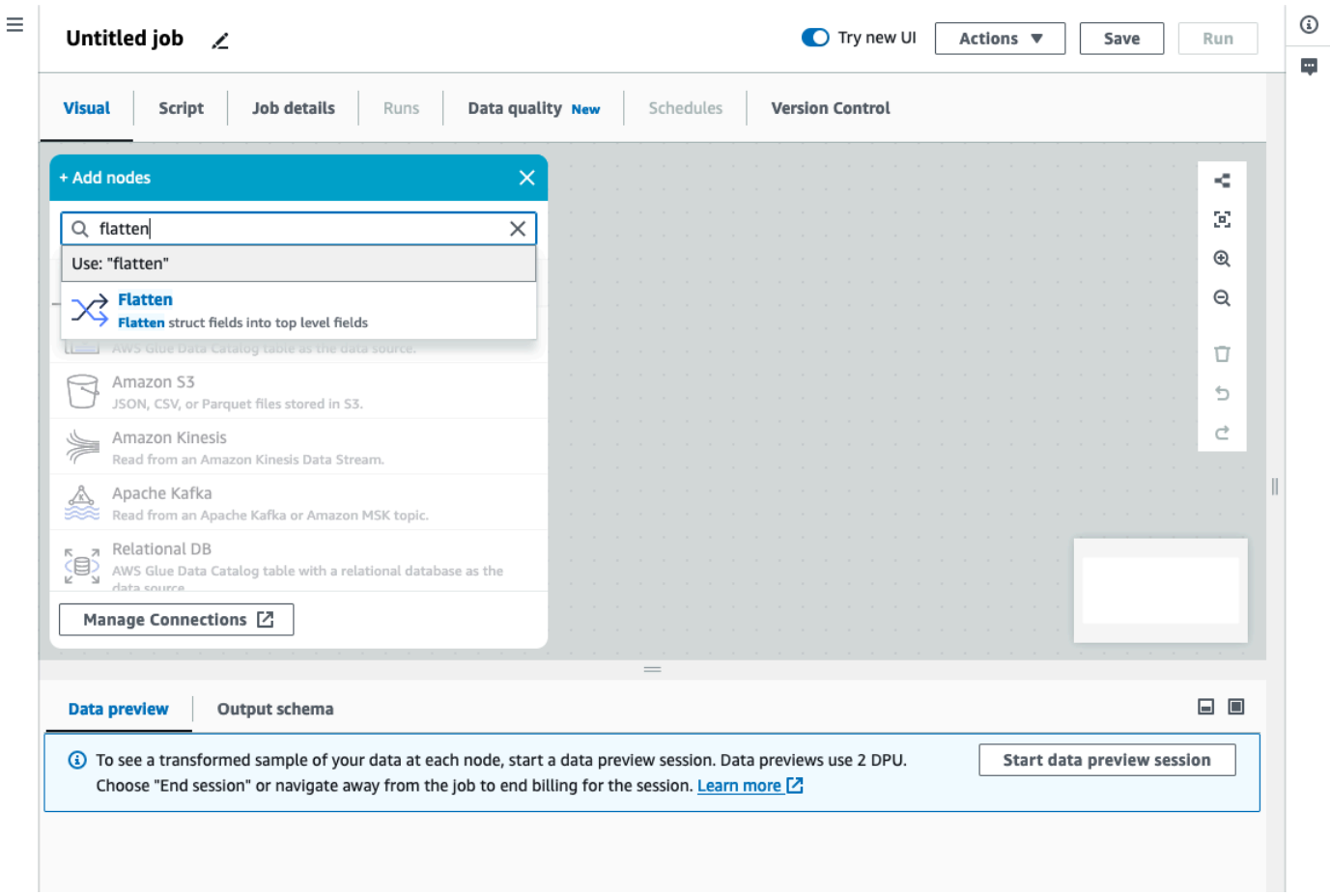
Nivele os campos de structs aninhados nos dados para transformá-los em campos de nível superior. Os novos campos são nomeados usando o nome do campo prefixado com os nomes dos campos de struct para alcançá-lo, separados por pontos.

Por exemplo, se os dados tiverem um campo do tipo Struct denominado “phone_numbers” que, entre outros campos, tenha um do tipo “Struct” denominado “home_phone” com dois campos: “country_code” e “number”. Depois de nivelados, esses dois campos se tornarão campos de nível superior denominados: “phone_numbers.home_phone.country_code” e “phone_numbers.home_phone.number”, respectivamente.

Para adicionar um nó de transformação Nivelar ao diagrama do trabalho

1. Abra o painel Recurso e escolha a guia Transformações e escolha Nivelar para adicionar uma nova transformação ao diagrama do trabalho. Você também pode usar a barra de pesquisa

digitando 'Flatten' e clicando no nó Nivelar. O nó selecionado no momento da adição do nó será o nó superior.



2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. (Opcional) Na guia Transformação, você pode limitar o nível máximo de aninhamento máximo a ser nivelado. Por exemplo, definir esse valor como 1 significa que somente structs de nível superior serão nivelados. Definir o máximo como 2 nivela o nível superior e os structs diretamente abaixo dele.

Adicionar uma coluna de UUID

Quando você adiciona uma coluna UUID (Universally Unique Identified), cada linha recebe uma string exclusiva de 36 caracteres.

Para adicionar um nó de transformação UUID ao diagrama do trabalho

1. Abra o painel Recurso e escolha UUID para adicionar uma nova transformação ao seu diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. (Opcional) Na guia Transformação, você pode personalizar o nome da nova coluna. Por padrão, ela será denominada "uuid".

Adicionar uma coluna de identificadores

Atribua um identificador numérico a cada linha no conjunto de dados.

Para adicionar um nó de transformação identificador ao diagrama do trabalho

1. Abra o painel Recurso e escolha Identificador para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. (Opcional) Na guia Transformação, você pode personalizar o nome da nova coluna. Por padrão, ela será denominada "id".
4. (Opcional) Se o trabalho processar e armazenar dados de forma incremental, será importante evitar que os mesmos IDs sejam reutilizados em diversas execuções do trabalho.

   Na guia Transformação, marque a opção exclusivo da caixa de seleção. Essa opção incluirá o timestamp do trabalho no identificador, tornando-o exclusivo entre as diversas execuções. Para permitir um número maior, a coluna em vez do tipo longo será um decimal.

Converter uma coluna no tipo timestamp

Você pode usar a transformação Em timestamp para alterar o tipo de dados de uma coluna de numérico ou string em timestamp, para que ela possa ser armazenada com esse tipo de dados ou aplicada a outras transformações que exijam um timestamp.

Para adicionar um nó de transformação Em timestamp ao diagrama do trabalho

1. Abra o painel Recurso e escolha Em Para timestamp para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformação, insira o nome da coluna a ser convertida.
4. Na guia Transformação, defina como analisar a coluna selecionada escolhendo o tipo.

   Se o valor for um número, ele poderá ser expresso em segundos (timestamp do Unix/Python), milissegundos ou microssegundos; escolha a opção correspondente.

   Se o valor for uma string formatada, escolha o tipo “iso”, e a string precisará estar conforme com uma das variantes do formato ISO, por exemplo: “2022-11-02T14:40:59.915Z“.

   Se não souber o tipo nesse momento ou se linhas diferentes usarem tipos diferentes, você poderá escolher ”detecção automática“ e o sistema fará sua melhor suposição, com um pequeno custo de desempenho.
5. (Opcional) Na guia Transformação, em vez de converter a coluna selecionada, você pode criar uma nova e manter a original inserindo um nome para a nova coluna.

Converter uma coluna de timestamp em uma string formatada

Formate uma coluna do tipo timestamp em uma do tipo string de acordo com um padrão. Você pode usar Formatar timestamp para obter a data e a hora como uma string com o formato desejado. Você pode definir o formato usando a sintaxe de data do Spark, bem como a maioria dos códigos de data do Python.

Por exemplo, se você quiser que a string de data seja formatada como “2023-01-01 00:00”, poderá definir esse formato usando a sintaxe do Spark como “aaaa-MM-dd HH:mm” ou os códigos de data equivalentes do Python como “%A-%m-%d %H: %M”

Para adicionar um nó de transformação Formatar timestamp ao diagrama do trabalho

1. Abra o painel Recurso e escolha Formatar timestamp para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.

2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformação, insira o nome da coluna a ser convertida.
4. Na guia Transformação, insira o padrão de Formato de timestamp a ser usado, expresso com a sintaxe de data do Spark ou os códigos de data do Python.
5. (Opcional) Na guia Transformação, em vez de converter a coluna selecionada, você pode criar uma nova e manter a original inserindo um nome para a nova coluna.

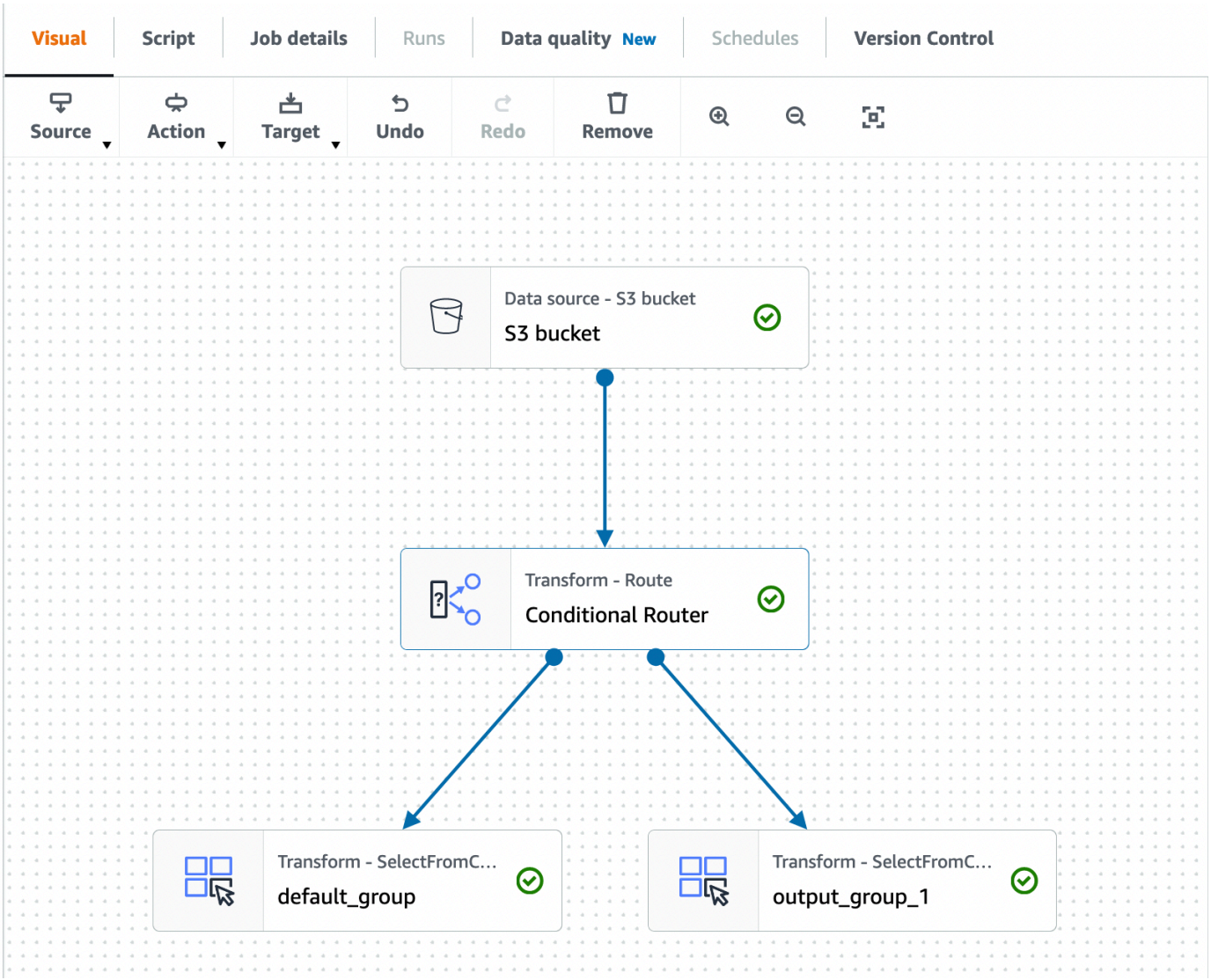
Criar uma transformação de roteador condicional

A transformação de roteador condicional permite que você aplique várias condições aos dados de entrada. Cada linha dos dados de entrada avaliada por uma condição de filtro de grupo e processada no grupo correspondente. Se uma linha atender a mais de uma condição de filtro de grupo, a transformação passará a linha para vários grupos. Se uma linha não atender a nenhuma condição, poderá ser descartada ou roteada para um grupo de saída padrão.

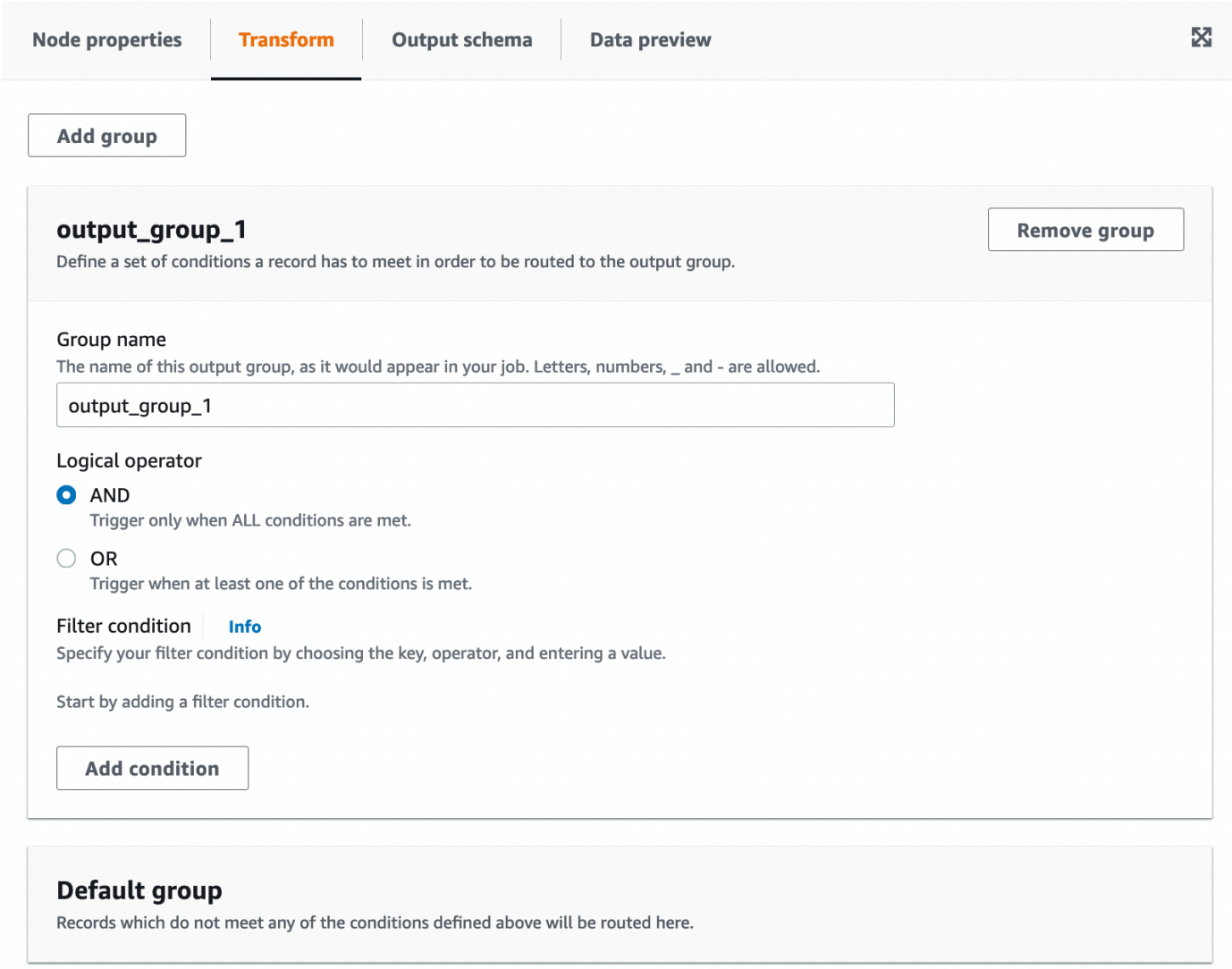
Essa transformação é semelhante à transformação de filtro, mas é útil para usuários que desejem testar os mesmos dados de entrada em várias condições.

Para adicionar uma transformação condicional de roteador:

1. Escolha um nó onde você realizará a transformação de roteador condicional. Esse pode ser um nó de origem ou outra transformação.
2. Escolha Ação e use a barra de pesquisa para localizar e escolher “Roteador condicional”. Uma transformação de roteador condicional é adicionada junto com dois nós de saída. Um único nó de saída, "Grupo padrão", contém os registros que não atendem a nenhuma das condições definidas nos outros nós de saída. O grupo padrão não pode ser editado.

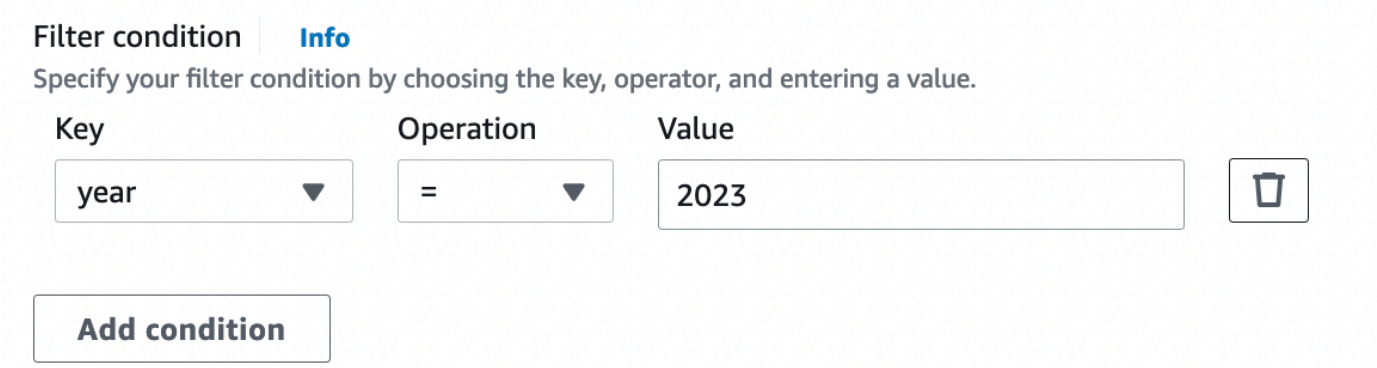


Você pode adicionar outros grupos de saída escolhendo Adicionar grupo. Para cada grupo de saída, você pode nomear o grupo e adicionar condições de filtro e um operador lógico.



3. Dê outro nome ao grupo de saída inserindo um novo nome para o grupo. O AWS Glue Studio nomeará automaticamente os grupos para você (por exemplo, "output_group_1").
4. Escolha um operador lógico (AND, OR) e adicione uma condição de filtro especificando Chave, Operação e Valor. Os operadores lógicos permitem que você implemente mais de uma condição de filtro e aplique o operador lógico a cada condição de filtro especificada.

   Ao especificar a chave, você pode escolher entre as chaves disponíveis em seu esquema. Em seguida, escolha a operação disponível de acordo com o tipo de chave selecionada. Por exemplo, se o tipo de chave for "string", a operação disponível para você escolher é "matches".



5. Insira o valor no campo Valor. Selecione Adicionar condição para adicionar mais condições. Para remover condições de filtro, selecione o ícone de lixeira.

Usando a transformação Concatenate Columns para acrescentar colunas

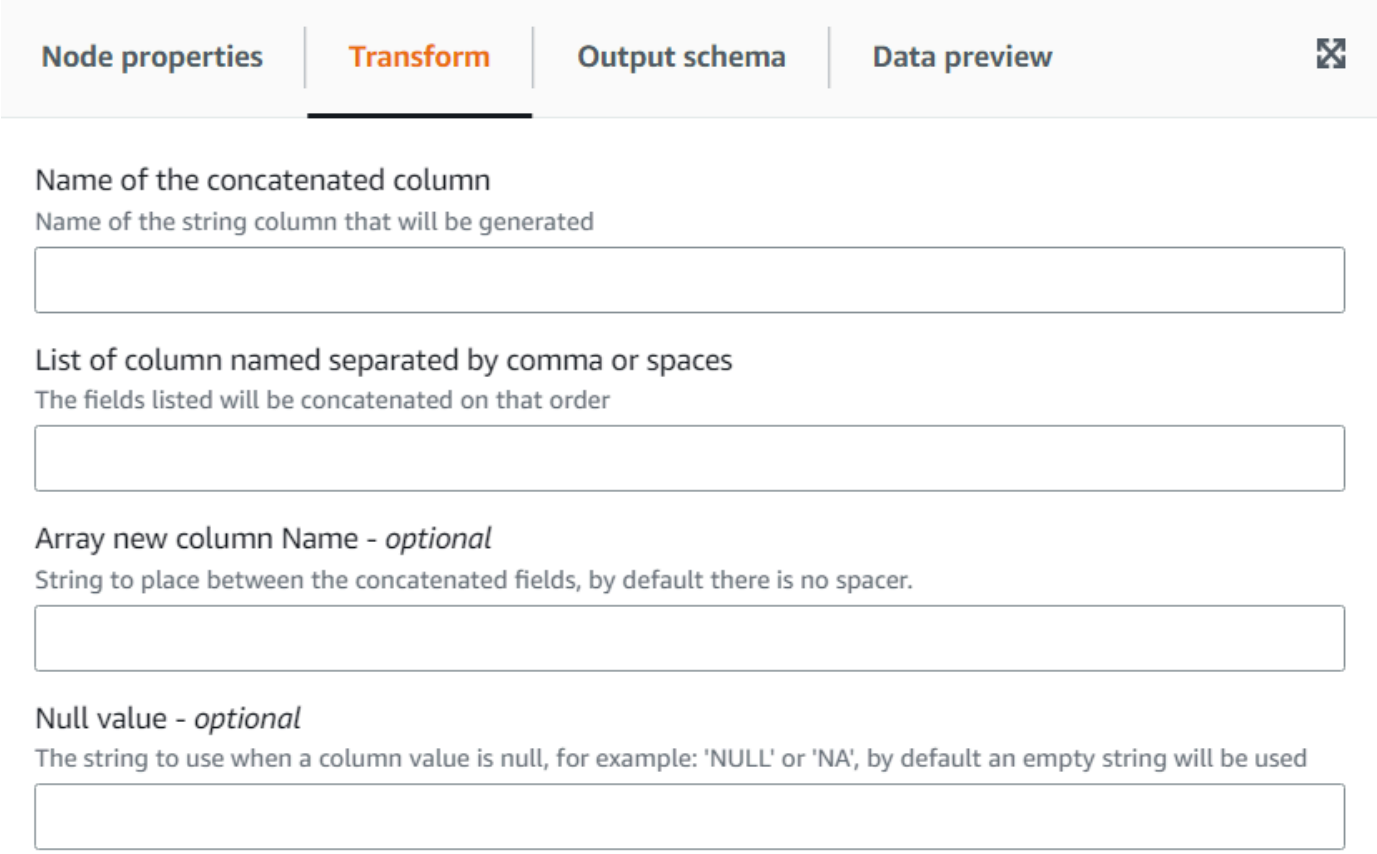
A transformação Concatenate permite que você crie uma nova coluna de strings usando os valores de outras colunas com um espaçador opcional. Por exemplo, se definirmos uma coluna concatenada “data” como a concatenação de “ano”, “mês” e “dia” (nessa ordem) com “-” como espaçador, obteríamos:

dia	mês	ano	data
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 adicionar uma transformação Concatenate:

1. Abra o painel Recurso. Em seguida, escolha Concatenate Columns para adicionar uma nova transformação ao diagrama de trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.

3. Na guia Transformar, insira o nome da coluna que conterá a string concatenada, bem como as colunas a serem concatenadas. A ordem na qual você marcar as colunas na lista suspensa será a ordem usada.



4. Espaçador - opcional - Insira uma string para colocar entre os campos concatenados. Por padrão, não há espaçador.
5. Valor nulo - opcional: insira uma string a ser usada quando o valor de uma coluna for nulo. Por padrão, nos casos em que as colunas têm o valor 'NULL' ou 'NA', é usada uma string vazia.

Usar a transformação Split String para dividir uma coluna de string

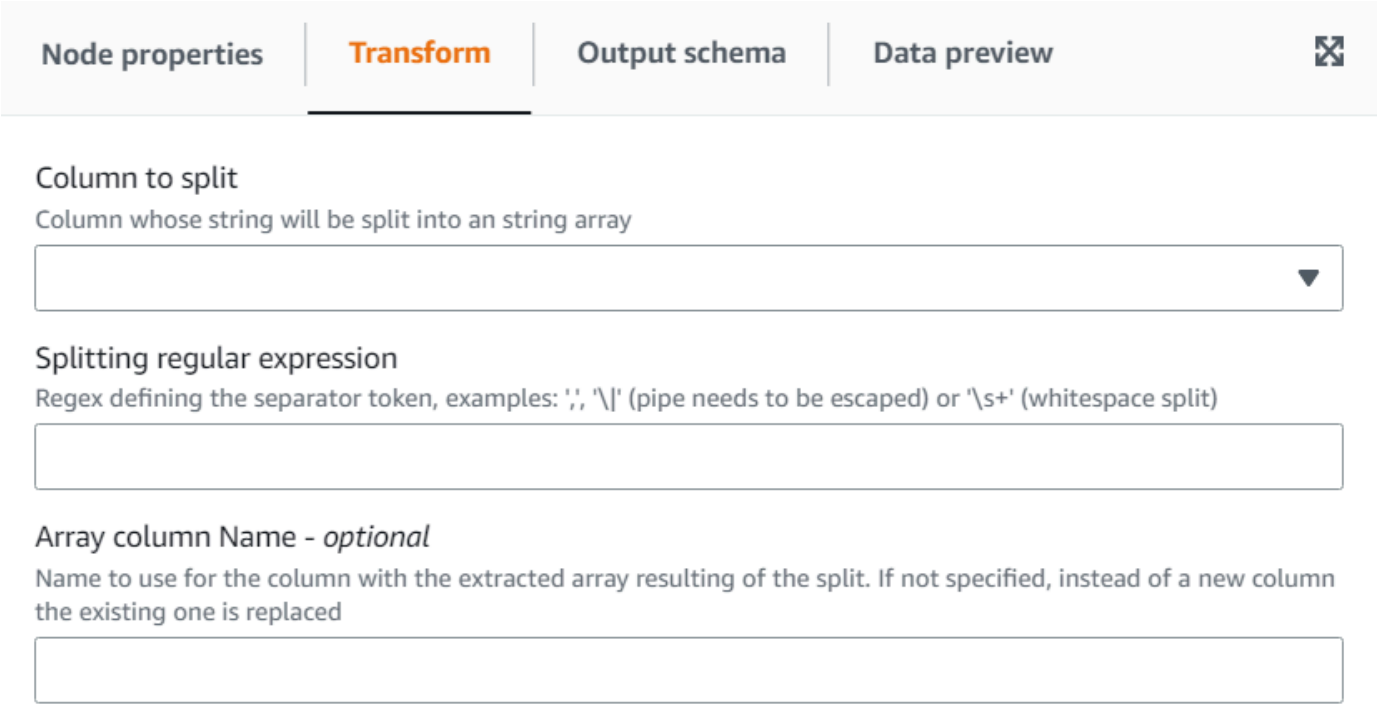
A transformação Split String permite que você divida uma string em uma matriz de tokens usando uma expressão regular para definir como a divisão é feita. Em seguida, você pode manter a coluna como um tipo de matriz ou aplicar uma transformação Array To Columns após esta, para extrair os valores da matriz nos campos de nível superior, supondo que cada token tenha um significado que conhecemos de antemão. Além disso, se a ordem dos tokens for irrelevante (por exemplo, um conjunto de categorias), você poderá usar a transformação Explode para gerar uma linha separada para cada valor.

Por exemplo, você pode dividir a coluna “categories” usando uma vírgula como padrão para adicionar uma coluna “categories_arr”.

product_id	categories	categories_arr
1	esportes, inverno	[esportes, inverno]
2	jardim, ferramentas	[jardim, ferramentas]
3	videogames	[videogames]
4	jogo, jogo de tabuleiro, social	[jogo, jogo de tabuleiro, social]

Para adicionar uma transformação Split String:

1. Abra o painel Recurso e escolha Split String para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformar, escolha a coluna a ser dividida e insira o padrão a ser usado para dividir a string. Na maioria dos casos, você pode simplesmente inserir o(s) caractere(s), a menos que tenha um significado especial como expressão regular e precise de escape. Os caracteres que precisam de escape são: `\.[]{}()<>*+-=!?^$|` adicionando uma barra invertida na frente do caractere. Por exemplo, se você quiser separar por um ponto ('.'), você precisa inserir `\.`. No entanto, uma vírgula não tem um significado especial e só pode ser especificada como está: `,`.



4. (Opcional) Se você quiser manter a coluna de string original, poderá inserir um nome para uma nova coluna de matriz, mantendo assim a coluna de string original e a nova coluna de matriz tokenizada.

Usando a transformação Array To Columns para extrair os elementos de uma matriz para colunas de nível superior

A transformação Array To Columns permite extrair alguns ou todos os elementos de uma coluna do tipo matriz para novas colunas. A transformação preencherá as novas colunas o máximo possível se a matriz tiver valores suficientes para extrair, opcionalmente usando os elementos nas posições especificadas.

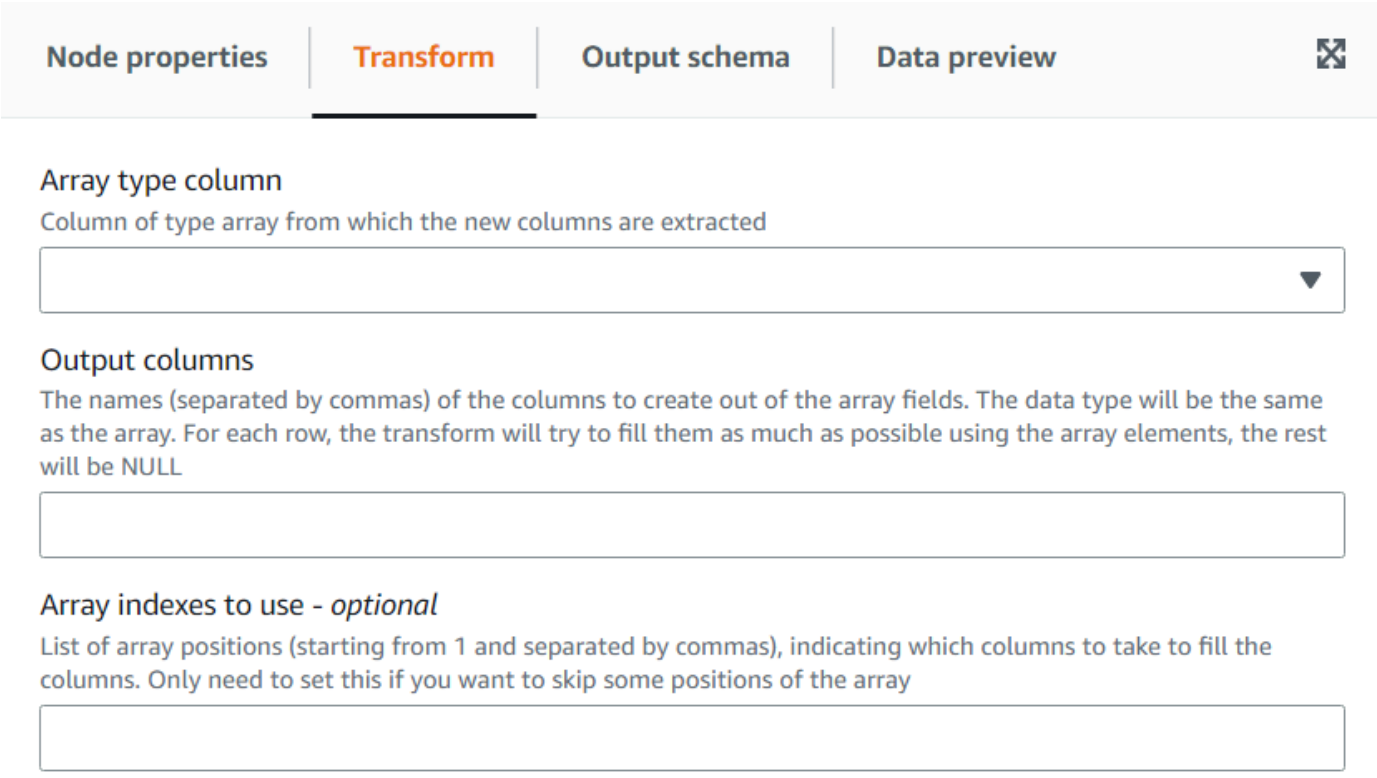
Por exemplo, se você tiver uma coluna de matriz “sub-rede”, que foi o resultado da aplicação da transformação “Split String” em uma sub-rede ip v4, você pode extrair a primeira e a quarta posições para as novas colunas “primeiro_octeto” e “quarto_octeto”. A saída da transformação neste exemplo seria (observe que as duas últimas linhas têm matrizes mais curtas do que o esperado):

sub-rede	primeiro_octeto	quarto_octeto
[54, 240, 197, 238]	54	238

sub-rede	primeiro_octeto	quarto_octeto
[192, 168, 0, 1]	192	1
[192, 168]	192	
[]		

Para adicionar uma transformação Array To Columns:

1. Abra o painel Recurso e escolha Array To Columns para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformar, escolha a coluna da matriz a ser extraída e insira a lista de novas colunas para os tokens extraídos.



4. (Opcional) Se não quiser usar os tokens da matriz para atribuir às colunas, você pode especificar os índices a serem obtidos, que serão atribuídos à lista de colunas na mesma ordem

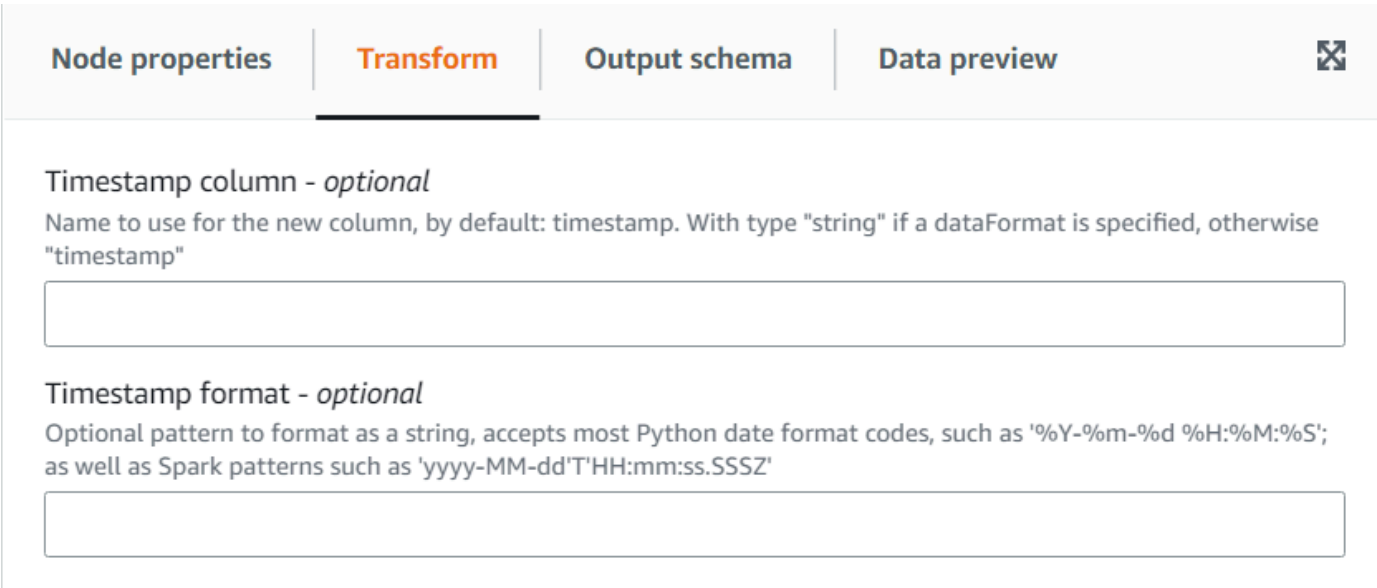
especificada. Por exemplo, se as colunas de saída forem “coluna1, coluna2, coluna3” e os índices “4, 1, 3”, o quarto elemento da matriz irá para a coluna1, o primeiro para a coluna2 e o terceiro para a coluna3 (se a matriz for menor que o número do índice, será atribuído um valor NULL).

Usar a transformação Add Current Timestamp

A transformação Add Current Timestamp permite marcar as linhas com a hora em que os dados foram processados. Isso é útil para fins de auditoria ou para rastrear a latência no pipeline de dados. Você pode adicionar essa nova coluna como um tipo de dados de carimbo de data/hora ou como string formatada.

Para adicionar uma transformação Add Current Timestamp:

1. Abra o painel Recurso e escolha Add Current Timestamp para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.



3. (Opcional) Na guia Transformar, insira um nome personalizado para a nova coluna e um formato se você preferir que a coluna seja uma string de data formatada.

Usar a transformação Pivot Rows to Columns

A transformação Pivot Rows to Columns permite agregar uma coluna numérica girando valores exclusivos nas colunas selecionadas que se tornam novas colunas (se várias colunas forem selecionadas, os valores serão concatenados para nomear as novas colunas). Dessa forma, as linhas são consolidadas e têm mais colunas com agregações parciais para cada valor exclusivo. Por exemplo, se você tiver esse conjunto de dados de vendas por mês e país (classificado para ser mais fácil de ilustrar):

ano	mês	país	valor
2020	Jan	uk	32
2020	Jan	de	42
2020	Jan	us	64
2020	Fev	uk	67
2020	Fev	de	4
2020	Fev	de	7
2020	Fev	us	6
2020	Fev	us	12
2020	Jan	us	90

Se você utilizar valor e país como as colunas de agregação, novas colunas serão criadas a partir da coluna país original. Na tabela abaixo, você tem novas colunas para de, uk e us em vez da coluna país.

ano	mês	de	uk	us
2020	Jan	42	32	64
2020	Jan	11	67	18

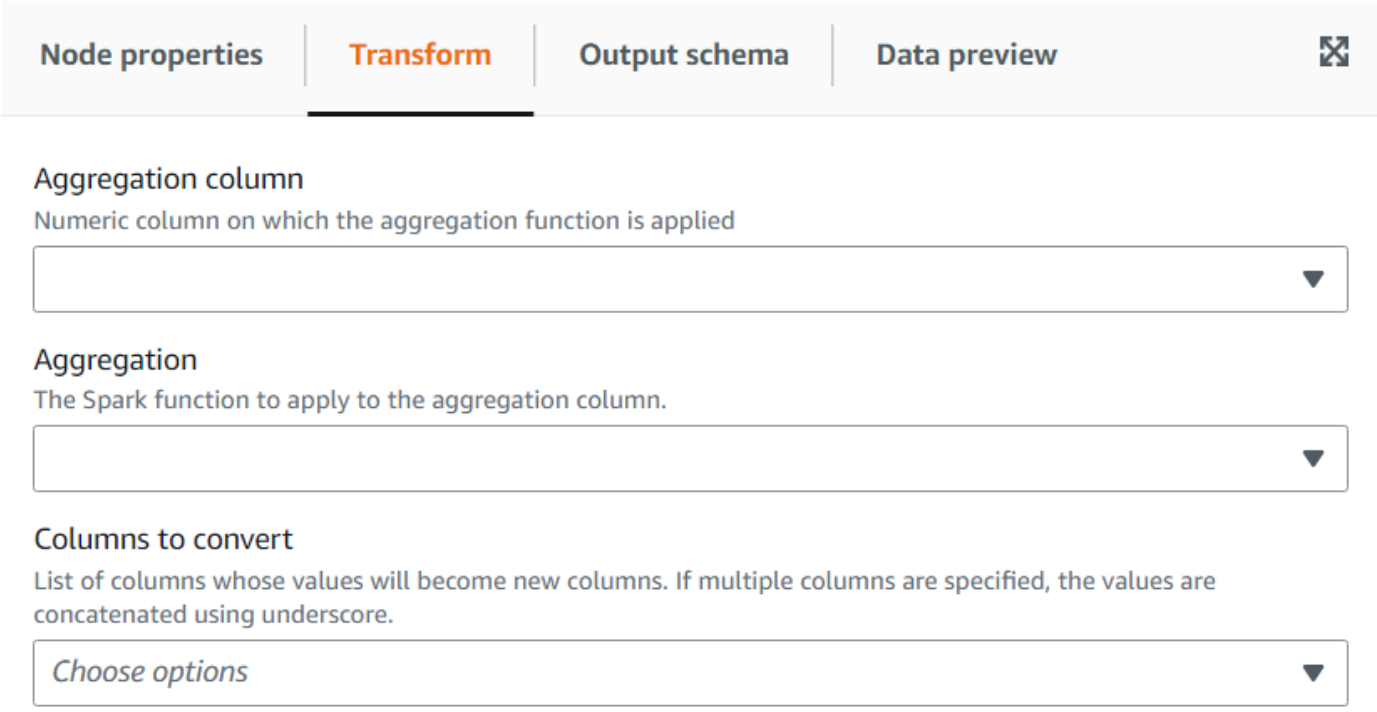
ano	mês	de	uk	us
2021	Jan			90

Se, em vez disso, você quiser dinamizar o mês e o país, obterá uma coluna para cada combinação dos valores dessas colunas:

ano	Jan_de	Jan_uk	Jan_us	Fev_de	Fev_uk	Fev_us
2020	42	32	64	11	67	18
2021			90			

Para adicionar uma transformação Pivot Rows Ro Columns:

1. Abra o painel Recurso e escolha Pivot Rows To Columns para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformar, escolha a coluna numérica que será agregada para produzir os valores para as novas colunas, a função de agregação a ser aplicada e a(s) coluna(s) cujos valores exclusivos serão convertidos em novas colunas.



Usar a transformação Unpivot Columns To Rows

A transformação Unpivot permite converter colunas em valores de novas colunas, gerando uma linha para cada valor exclusivo. É o oposto de pivot, mas observe que não é equivalente, pois não pode separar linhas com valores idênticos que eram agregados ou dividir combinações nas colunas originais (você pode fazer isso posteriormente usando uma transformação Split). Por exemplo, se tiver a seguinte tabela:

ano	mês	de	uk	us
2020	Jan	42	32	64
2020	Fev	11	67	18
2021	Jan			90

Você pode despivotar as colunas: “de”, “uk” e “us” em uma coluna “país” com o valor “quantidade” e obter o seguinte (classificado aqui para fins de ilustração):

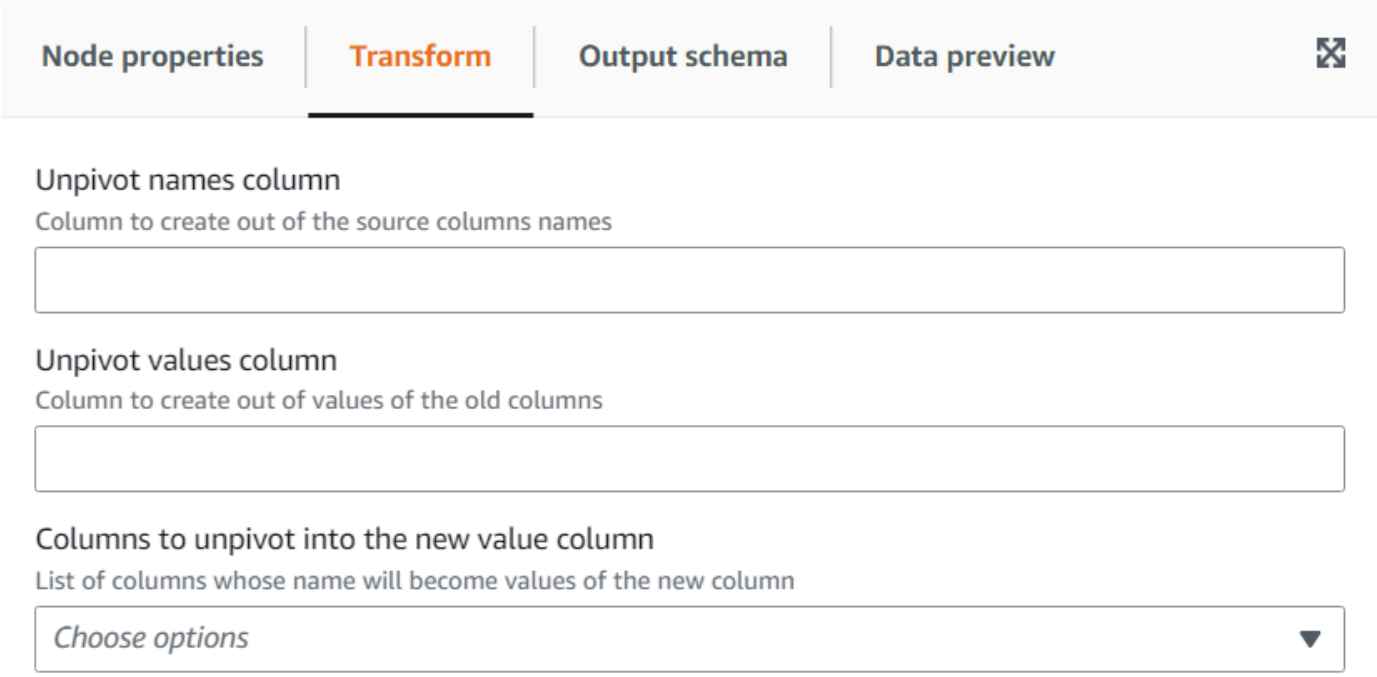
ano	mês	país	valor
2020	Jan	uk	32
2020	Jan	de	42
2020	Jan	us	64
2020	Fev	uk	67
2020	Fev	de	11
2020	Fev	us	18
2021	Jan	us	90

Observe que as colunas que têm um valor NULL (“de” e “uk” de jan 2021) não são geradas por padrão. Você pode ativar essa opção para obter:

ano	mês	país	valor
2020	Jan	uk	32
2020	Jan	de	42
2020	Jan	us	64
2020	Fev	uk	67
2020	Fev	de	11
2020	Fev	us	18
2021	Jan	us	90
2021	Jan	de	
2021	Jan	uk	

Para adicionar uma transformação Unpivot Columns to Rows:

1. Abra o painel Recurso e escolha Unpivot Rows To Columns para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformar, insira as novas colunas a serem criadas para conter os nomes e valores das colunas escolhidas para despivotar.

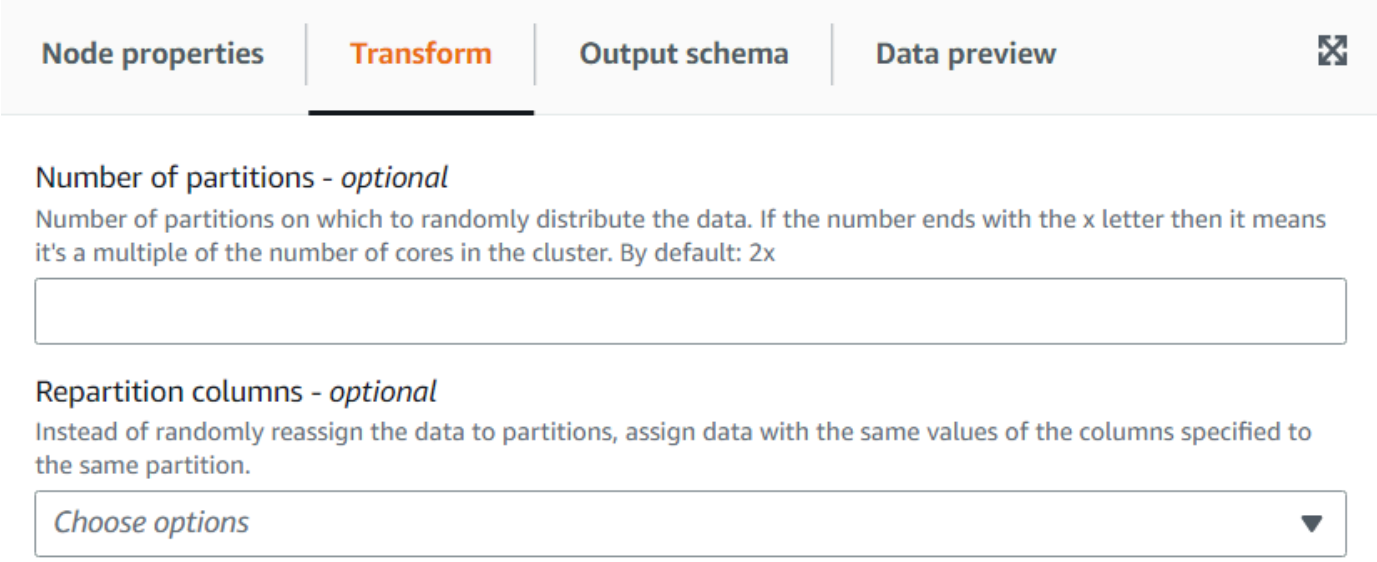


Usando a transformação Autobalance Processing para otimizar seu runtime

A transformação Autobalance Processing redistribui os dados entre os operadores para melhorar a performance. Isso ajuda nos casos em que os dados estão desbalanceados ou, como vêm da fonte, não permitem processamento paralelo suficiente. Isso é comum quando a fonte é compactada com gzip ou é JDBC. A redistribuição de dados tem um custo de performance módico, portanto, a otimização nem sempre compensará esse esforço se os dados já estiverem bem balanceados. Por baixo, a transformação usa a repartição do Apache Spark para reatribuir dados aleatoriamente entre um número de partições ideal para a capacidade do cluster. Para usuários avançados, é possível inserir um número de partições manualmente. Além disso, ele pode ser usado para otimizar

a gravação de tabelas particionadas reorganizando os dados com base em colunas especificadas. Isso resulta em arquivos de saída mais consolidados.

1. Abra o painel Recurso e escolha Autobalance Processing para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. (Opcional) Na guia Transformação, você pode inserir um número de partições. Em geral, é recomendável que você deixe o sistema decidir esse valor, mas pode ajustar o multiplicador ou inserir um valor específico se precisar controlar isso. Se você for salvar os dados particionados por colunas, poderá escolher as mesmas colunas que as colunas de repartição. Dessa forma, minimizará o número de arquivos em cada partição e evitará muitos arquivos por partição, o que prejudicaria a performance das ferramentas que consultam esses dados.



Usando a transformação Derived Column para combinar outras colunas

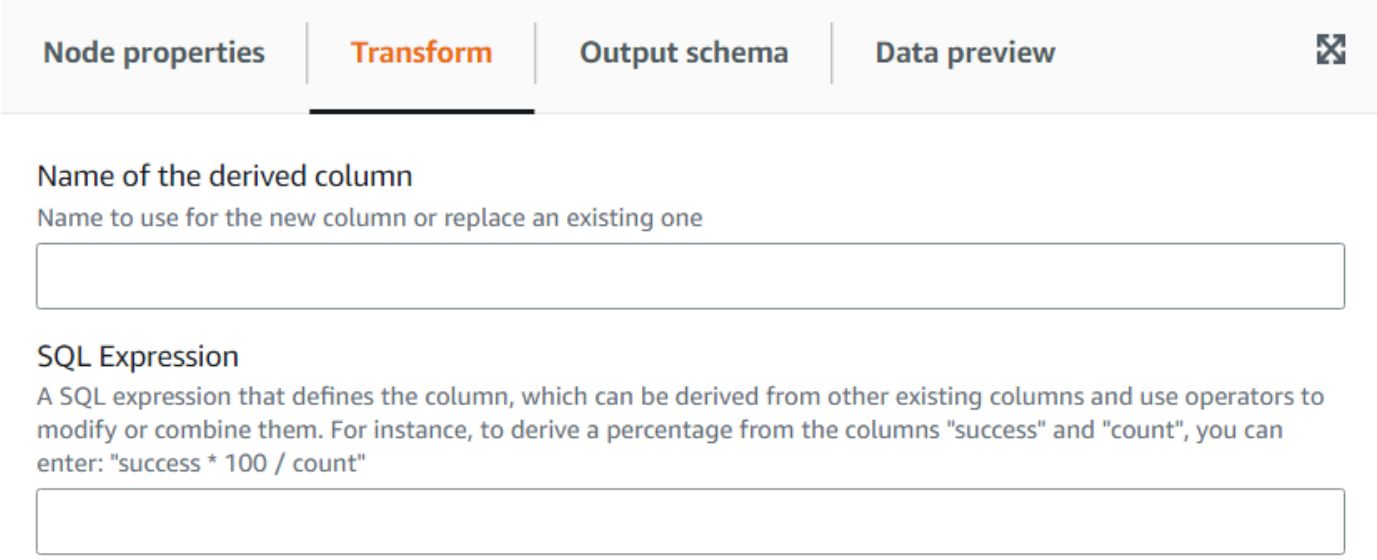
A transformação Derived Column permite definir uma nova coluna com base em uma fórmula matemática ou expressão SQL na qual você pode usar outras colunas nos dados, além de constantes e literais. Por exemplo, para derivar uma coluna de “porcentagem” das colunas “success” e “count”, você pode inserir a expressão SQL: “success * 100/count || '%'”.

Exemplo de resultado:

success	contagem	percentage
14	100	14%
6	20	3%
3	40	7,5%

Para adicionar uma transformação Derived Column:

1. Abra o painel Recurso e escolha Derived Column para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transform, insira o nome da coluna e a expressão do conteúdo.



Usar a transformação Lookup para adicionar dados correspondentes de uma tabela de catálogo

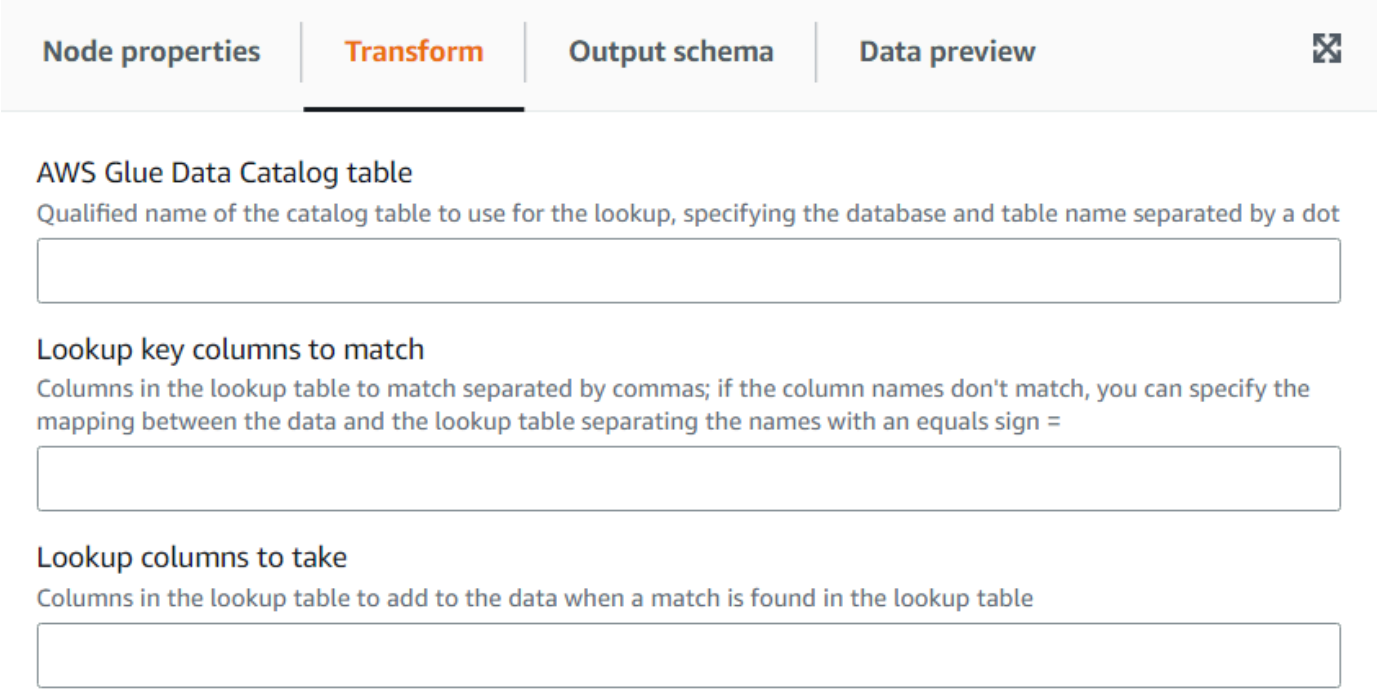
A transformação Lookup permite adicionar colunas de uma tabela de catálogo definida quando as chaves correspondem às colunas de pesquisa definidas nos dados. Isso equivale a fazer uma junção externa esquerda entre os dados e a tabela de pesquisa usando como condição as colunas correspondentes.

Para adicionar uma transformação Lookup:

1. Abra o painel Recurso e escolha Lookup para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformar, insira o nome totalmente qualificado da tabela de catálogo a ser usado para realizar as pesquisas. Por exemplo, se seu banco de dados for “mydb” e sua tabela “mytable”, digite “mydb.mytable”. Em seguida, insira os critérios para encontrar uma correspondência na tabela de pesquisa, se a chave de pesquisa for composta. Insira a lista de colunas de chave separadas por vírgulas. Se uma ou mais das colunas de chave não tiverem o mesmo nome, você precisará definir o mapeamento de correspondência.

   Por exemplo, se as colunas de dados forem “user_id” e “region” e na tabela de usuários as colunas correspondentes forem denominadas “id” e “region”, no campo Colunas a serem correspondidas, digite: "user_id=id, region". Você poderia fazer region=region, mas não é necessário, pois são iguais.
4. Por fim, insira as colunas a serem extraídas da linha correspondente na tabela de pesquisa para incorporá-las aos dados. Se nenhuma correspondência for encontrada, essas colunas serão definidas como NULL.

> **Note**
>
> Abaixo da transformação Lookup, está sendo usada uma junção à esquerda para ser eficiente. Se a tabela de pesquisa tiver uma chave composta, certifique-se de que as colunas correspondentes estejam configuradas para corresponder a todas as colunas-chave, de forma que somente uma correspondência possa ocorrer. Caso contrário, várias linhas de pesquisa corresponderão e isso resultará na adição de linhas extras para cada uma dessas correspondências.



Usando a transformação Explode Array ou Map Into Rows

A transformação Explode permite extrair valores de uma estrutura aninhada em linhas individuais que são mais fáceis de manipular. No caso de uma matriz, a transformação gerará uma linha para cada valor da matriz, replicando os valores para as outras colunas na linha. No caso de um mapa, a transformação gerará uma linha para cada entrada com a chave e o valor como colunas, além de quaisquer outras colunas na linha.

Por exemplo, se tivermos esse conjunto de dados que tem uma coluna de matriz “categoria” com vários valores.

product_id	category
1	[esportes, inverno]
2	[jardim, ferramentas]
3	[videogames]
4	[jogo, jogo de tabuleiro, social]

product_id	category
5	[]

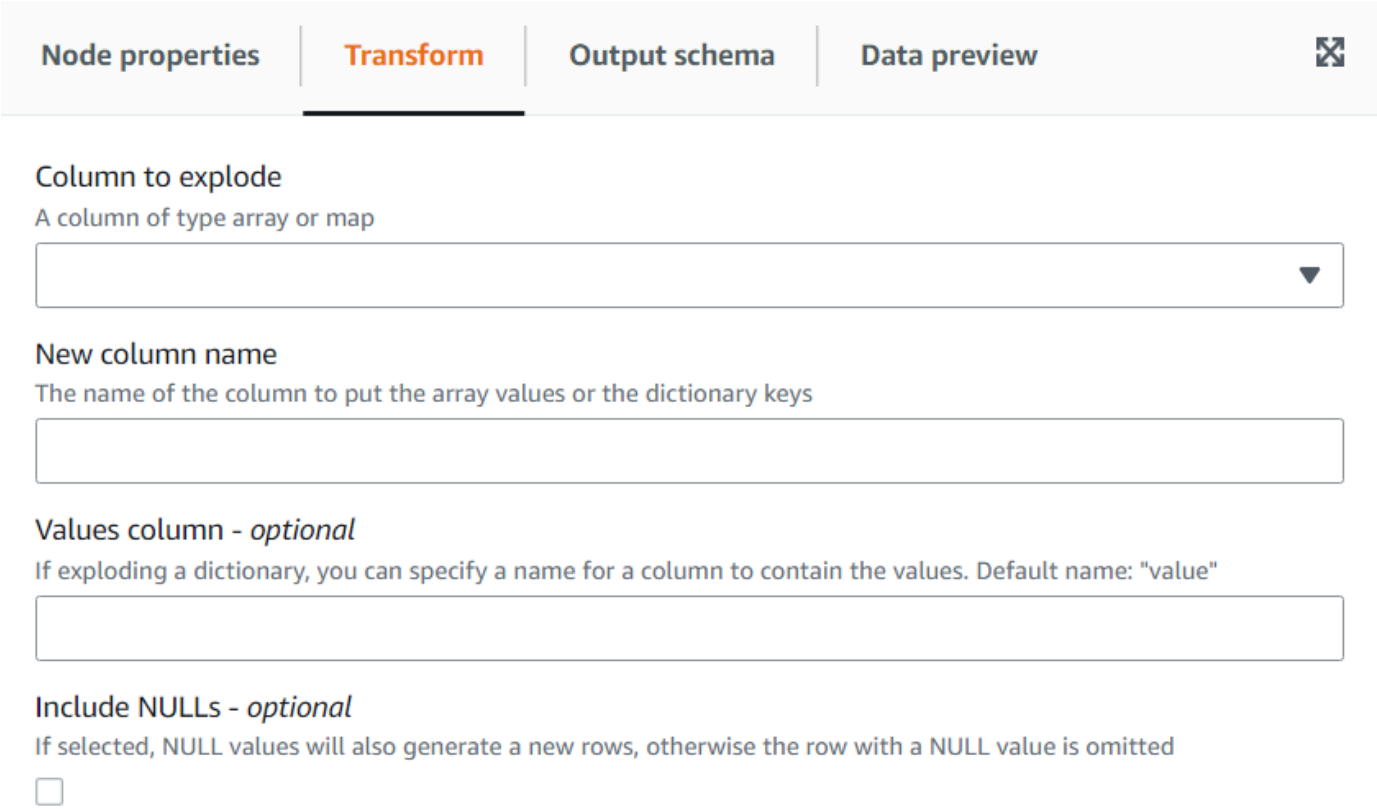
Se você explodir a coluna “categoria” em uma coluna com o mesmo nome, você substituirá a coluna. Você pode selecionar que deseja que os NULLs sejam incluídos para obter o seguinte (ordenado para fins ilustrativos):

product_id	category
1	esportes
1	inverno
2	jardim
2	ferramenta
3	videogames
4	jogo
4	jogo de tabuleiro
4	social
5	

Para adicionar uma transformação Explode Array ou Map Into Rows:

1. Abra o painel Recurso e escolha Explode Array Or Map Into Rows para adicionar uma nova transformação ao diagrama do seu trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. (Opcional) Na guia Propriedades do nó, insira um nome para o nó no diagrama do trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.

3. Na guia Transformar, escolha a coluna a ser explodida (ela deve ser um tipo de matriz ou mapa). Em seguida, insira um nome para a coluna para os itens da matriz ou os nomes das colunas para as chaves e valores se você estiver explodindo um mapa.
4. (Opcional) Na guia Transformar, por padrão, se a coluna a ser explodida for NULL ou tiver uma estrutura vazia, ela será omitida no conjunto de dados explodido. Se você quiser manter a linha (com as novas colunas como NULL), marque “Incluir NULLs”.

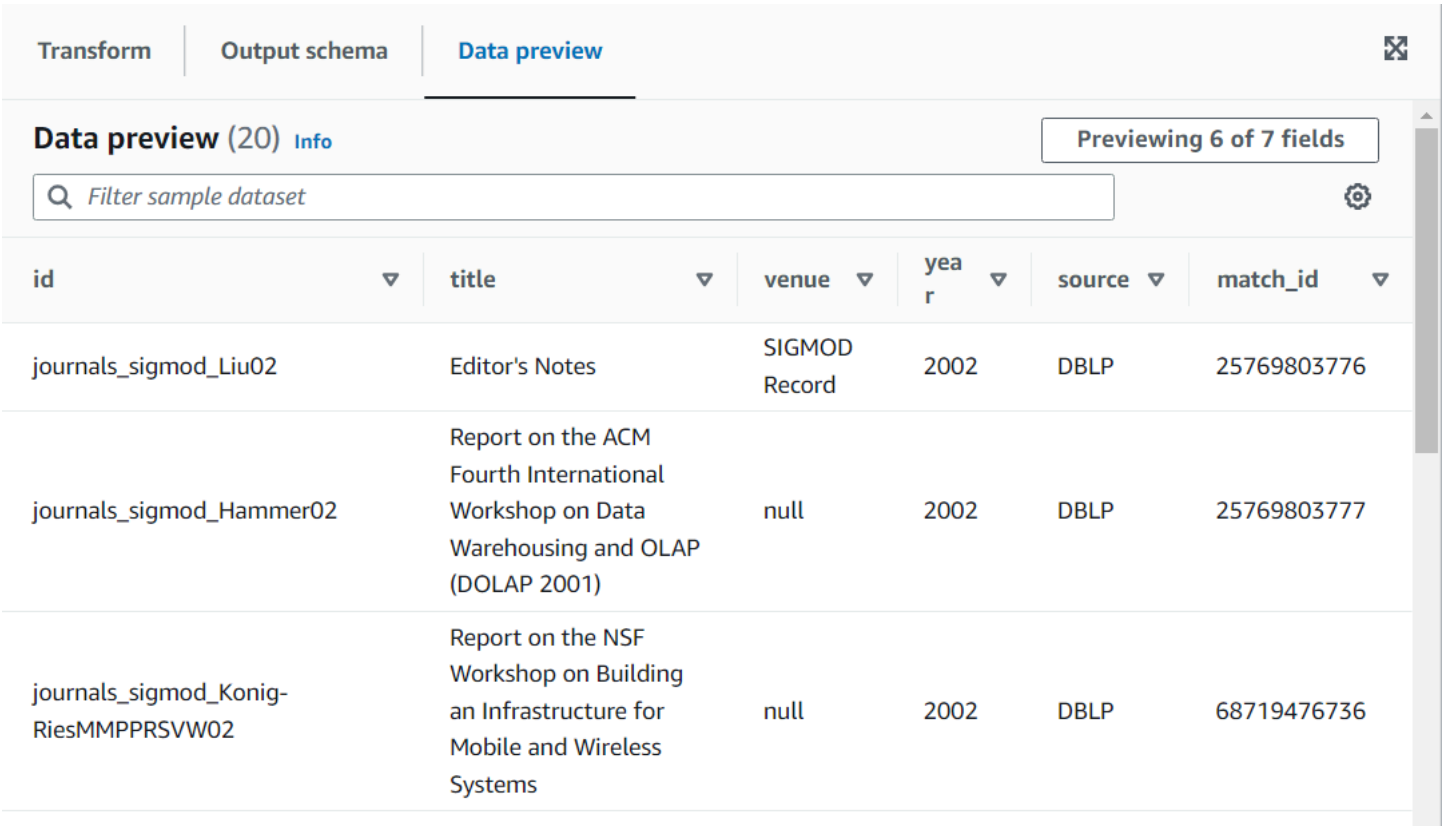


Usar a transformação Record Matching para invocar uma transformação de classificação de dados existentes

Usar a transformação invoca uma transformação de classificação de dados de machine learning Record Matching existente.

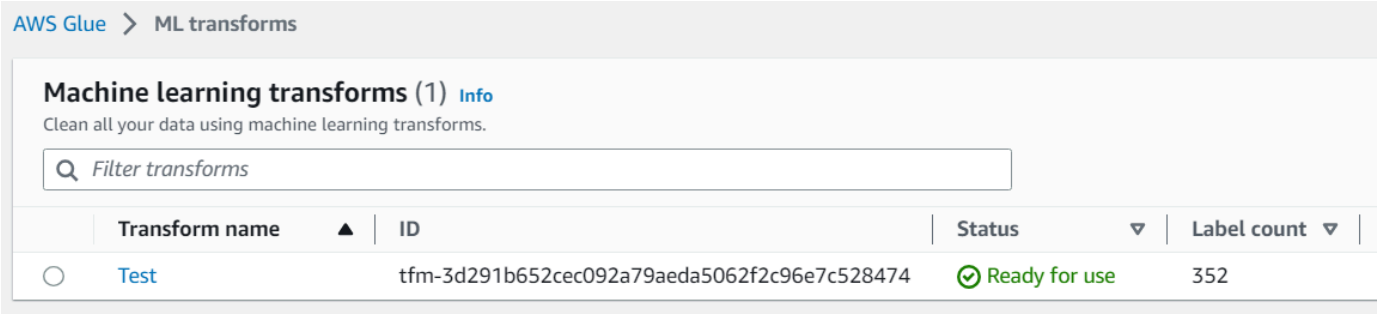
A transformação avalia os dados atuais em relação ao modelo treinado com base em rótulos. Uma coluna “match_id” é adicionada para atribuir cada linha a um grupo de itens considerados equivalentes com base no treinamento do algoritmo. Para obter mais informações, consulte Correspondência de registros com Lake Formation FindMatches.

> **Note**
>
> A versão do AWS Glue usada pelo trabalho visual deve corresponder à versão que o AWS Glue usou para criar a transformação Record Matching.



Para adicionar um nó de transformação Record Matching ao diagrama de trabalho

1. Abra o painel Recurso e escolha Record Matching para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. No painel propriedades do nó, insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformar, insira a ID retirada da página Transformações de machine learning:



4. (Opcional) Na guia Transformação, você pode marcar a opção de adicionar as pontuações de confiança. Ao custo de computação extra, o modelo estimará uma pontuação de confiança para cada correspondência como uma coluna adicional.

Remoção de linhas nulas

Essa transformação remove do conjunto de dados as linhas que têm todas as colunas como nulas. Além disso, você pode estender esses critérios para incluir campos vazios, de modo a manter as linhas em que pelo menos uma coluna não esteja vazia.

Para adicionar um nó de transformação Remove Null Rows ao diagrama de trabalho

1. Abra o painel Recurso e escolha Remove Null Rows para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. No painel propriedades do nó, insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. (Opcional) Na guia Transformar, marque a opção Estendido se quiser exigir que as linhas não apenas não sejam nulas, mas também não estejam vazias. Dessa forma, cadeias de caracteres, matrizes ou mapas vazios serão considerados nulos para o propósito dessa transformação.

Analisar uma coluna de string contendo dados JSON

Essa transformação analisa uma coluna de string contendo dados JSON e a converte em uma estrutura ou coluna de matriz, dependendo se o JSON é um objeto ou uma matriz, respectivamente. Opcionalmente, você pode manter a coluna analisada e a original.

O esquema JSON pode ser fornecido ou inferido (no caso de objetos JSON), com amostragem opcional.

Para adicionar um nó de transformação de Parse JSON Column ao diagrama de trabalho

1. Abra o painel Recurso e escolha Parse JSON Column para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. No painel propriedades do nó, insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformar, selecione a coluna que contém a string JSON.
4. (Opcional) Na guia Transformar, insira o esquema que os dados JSON seguem usando a sintaxe SQL, por exemplo: “campo1 STRING, campo2 INT” no caso de um objeto ou “ARRAY<STRING>” no caso de uma matriz.

   No caso de uma matriz, o esquema é obrigatório, mas no caso de um objeto, se o esquema não for especificado, ele será inferido usando os dados. Para reduzir o impacto da inferência do esquema (especialmente em um conjunto de dados grande), você pode evitar ler os dados inteiros duas vezes inserindo uma proporção de amostras a serem usadas para inferir o esquema. Se o valor for menor que 1, a proporção correspondente de amostras aleatórias será usada para inferir o esquema. Se os dados forem confiáveis e o objeto for consistente entre as linhas, você poderá usar uma proporção pequena, como 0,1, para melhorar a performance.
5. (Opcional) Na guia Transformar, você pode inserir um novo nome de coluna se quiser manter a coluna de string original e a coluna analisada.

Extraindo um caminho JSON

Essa transformação extrai novas colunas de uma coluna de string JSON. Essa transformação é útil quando você precisa apenas de alguns elementos de dados e não quer importar todo o conteúdo do JSON para o esquema da tabela.

Para adicionar um nó de transformação Extract JSON Path ao diagrama do trabalho

1. Abra o painel Recurso e escolha Extract JSON Path para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. No painel propriedades do nó, insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformar, selecione a coluna que contém a string JSON. Insira uma ou mais expressões de caminho JSON separadas por vírgulas, cada uma fazendo referência a como

extrair um valor da matriz ou objeto JSON. Por exemplo, se a coluna JSON contivesse objetos com as propriedades “prop_1" e “prop2", você poderia extrair ambas especificando seus nomes “prop_1, prop_2".

Se o campo JSON tiver caracteres especiais, por exemplo, para extrair a propriedade desse JSON `{"a. a": 1}` você poderá usar o caminho `$['a. a']`. A exceção é a vírgula porque ela é reservada para caminhos separados. Em seguida, insira os nomes das colunas correspondentes para cada caminho, separados por vírgulas.

4. (Opcional) Na guia Transformar, você pode marcar a opção de descartar a coluna JSON depois de extraída. Isso faz sentido quando você não precisa do restante dos dados JSON depois de extrair as partes necessárias.

Extrair fragmentos de string usando uma expressão regular

Essa transformação extrai fragmentos de string usando uma expressão regular e cria uma nova coluna a partir dela, ou várias colunas se estiver usando grupos regex.

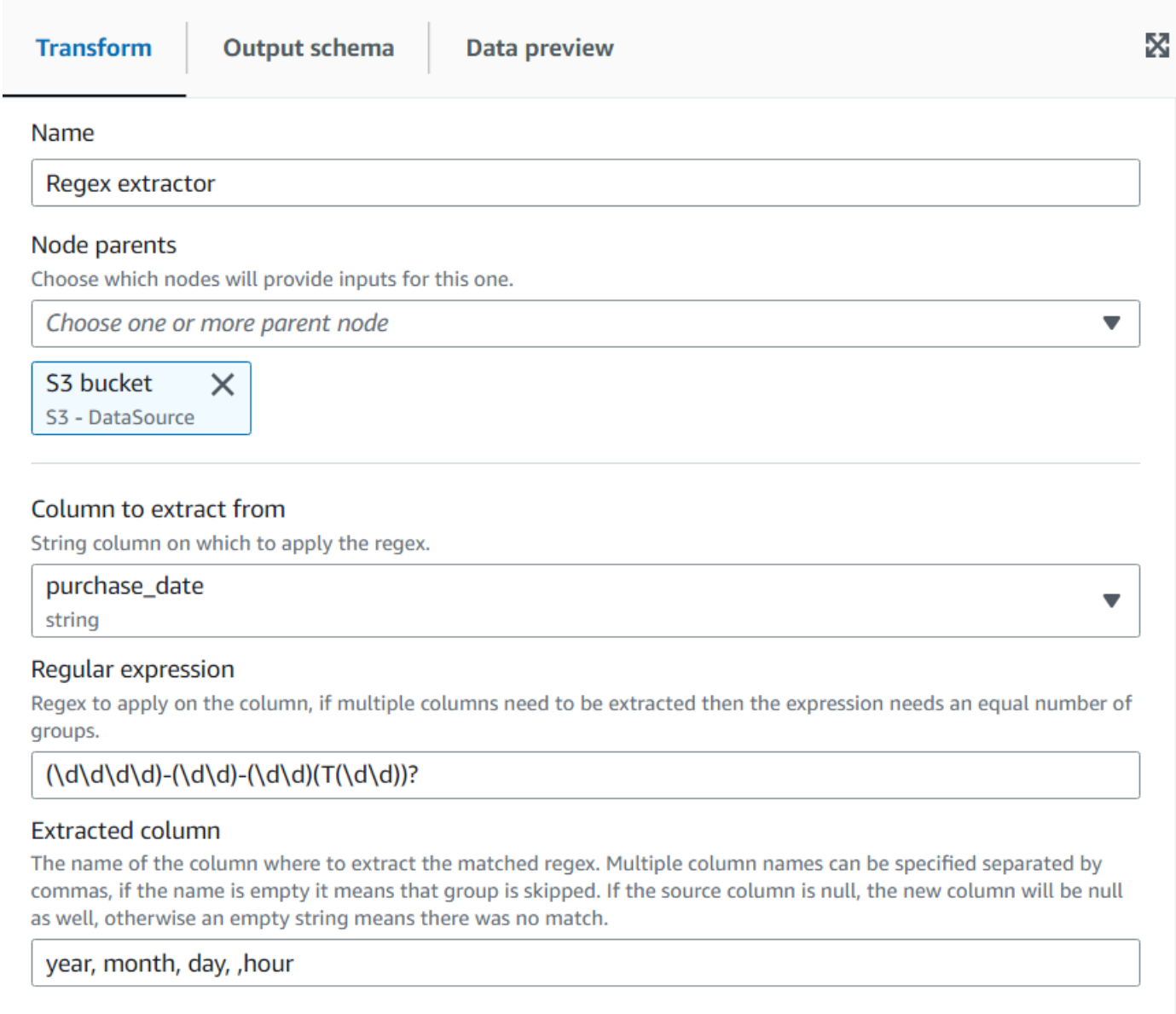
Para adicionar um nó de transformação Regex Extractor ao diagrama de trabalho

1. Abra o painel Recurso e escolha Regex Extractor para adicionar uma nova transformação ao diagrama do trabalho. O nó selecionado no momento da adição do nó será o nó superior.
2. No painel propriedades do nó, insira um nome para o nó no diagrama de trabalho. Se ainda não houver um nó pai selecionado, escolha um na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.
3. Na guia Transformar, insira a expressão regular e a coluna na qual ela precisa ser aplicada. Em seguida, insira o nome da nova coluna na qual armazenar a string correspondente. A nova coluna será nula somente se a coluna de origem for nula; se a regex não corresponder, a coluna ficará vazia.

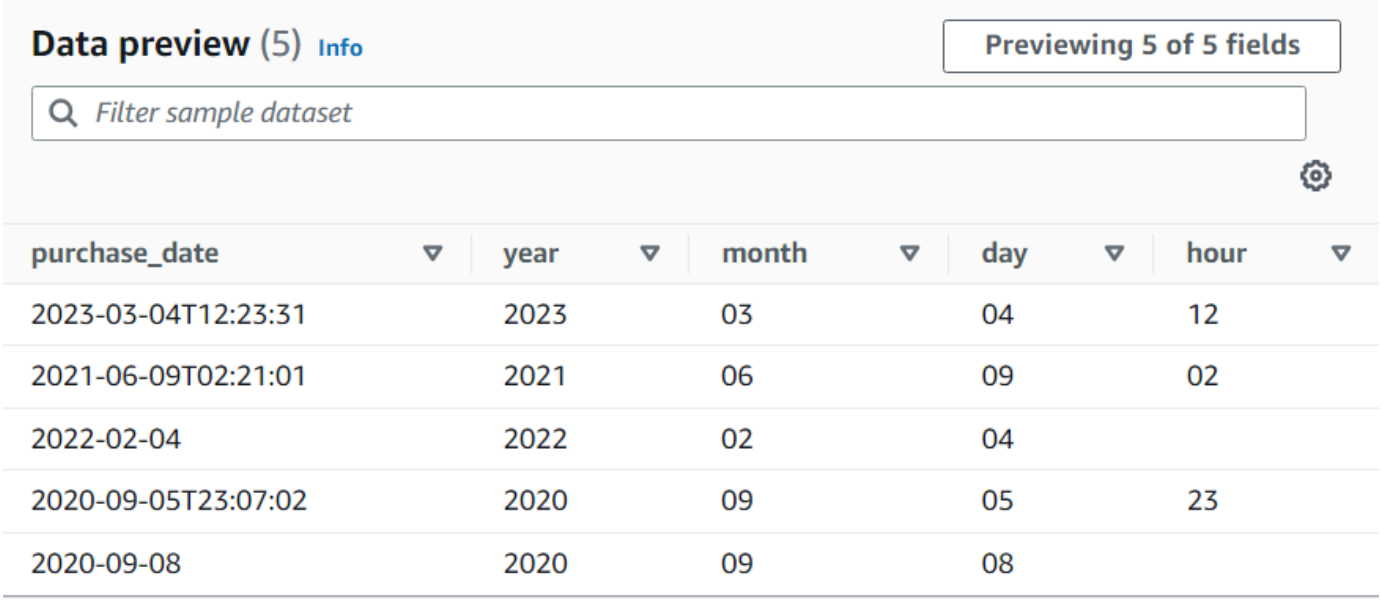
   Se a regex usar grupos, haverá um nome de coluna correspondente separado por vírgula, mas você pode pular grupos deixando o nome da coluna vazio.

   Por exemplo, se você tiver uma coluna “purchase_date” com uma string usando formatos de data ISO longo e curto, convém extrair o ano, mês, dia e hora, quando disponível. Observe que o grupo de horas é opcional, caso contrário, nas linhas em que não estiverem disponíveis, todos os grupos extraídos seriam strings vazias (porque o regex não correspondia). Nesse caso, não

queremos que o grupo torne a hora opcional, mas a interna, então deixamos o nome vazio e ele não é extraído (esse grupo incluiria o caractere T).



Resultando na visualização prévia dos dados:



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	

Criar uma transformação personalizada

Se precisar executar transformações mais complicadas em seus dados ou quiser adicionar chaves de propriedade de dados ao conjunto de dados, você pode adicionar uma transformação Custom code (Código personalizado) em seu diagrama de trabalho. O nó Custom code (Código personalizado) permite inserir um script que executa a transformação.

Ao usar código personalizado, você deve usar um editor de esquema para indicar as alterações feitas na saída por meio do código personalizado. Ao editar o esquema, você pode executar as seguintes ações:

- Adicionar ou remover chaves de propriedade de dados
- Alterar o tipo de dados das chaves de propriedade de dados
- Alterar o nome das chaves de propriedade de dados
- Reestruturar uma chave de propriedade aninhada

Você deve usar uma transformação SelectFromCollection (Selecionar da coleção) para escolher um único `DynamicFrame` do resultado do nó de transformação personalizada antes de enviar a saída para um local de destino.

Use as seguintes tarefas para adicionar um nó de transformação personalizada ao diagrama de trabalho.

Adicionar um nó de transformação de código personalizado ao diagrama de trabalho

Para adicionar um nó de transformação personalizado ao diagrama de trabalho

1. (Opcional) abra o painel Recurso e escolha Transformação personalizada para adicionar uma nova transformação ao diagrama de trabalho.
2. Na guia Node properties (Propriedades do nó), insira um nome para o nó no diagrama de trabalho. Se um nó pai ainda não estiver selecionado, ou se você quiser várias entradas para a transformação personalizada, escolha um nó na lista Node parents (Nós pais) para usar como fonte de entrada para a transformação.

Inserir código para o nó de transformação personalizada

Você pode digitar ou copiar código em um campo de entrada. O trabalho usa esse código para executar a transformação de dados. Você pode fornecer um trecho de código em Python ou Scala. O código deve aceitar um ou mais `DynamicFrames` como entrada e retornar uma coleção de `DynamicFrames`.

Para inserir o script para um nó de transformação personalizada

1. Com o nó de transformação personalizada selecionado no diagrama de trabalho, escolha a guia Transform (Transformação).
2. No campo de entrada de texto, no cabeçalho Code block (Bloco de código), cole ou insira o código da transformação. O código que você usa deve corresponder à linguagem especificada para o trabalho na guia Job details (Detalhes do trabalho).

   Ao fazer referência aos nós de entrada em seu código, o AWS Glue Studio nomeia o `DynamicFrames` retornado pelos nós do diagrama de trabalho sequencialmente com base na ordem de criação. Use um dos seguintes métodos de atribuição de nomes no código:

   - Geração de código clássico: use nomes funcionais para se referir aos nós no diagrama do trabalho.
     - Nós da origem dos dados: `DataSource0`, `DataSource1`, `DataSource2` e assim por diante.
     - Nós de transformação: `Transform0`, `Transform1`, `Transform2` e assim por diante.
   - Geração de novo código: use o nome especificado na guia Node properties (Propriedades do nó), seguida por um "`_node1`", "`_node2`" anexao e assim por

diante. Por exemplo, `S3bucket_node1`, `ApplyMapping_node2`, `S3bucket_node2`, `MyCustomNodeName_node1`.

Para obter mais informações sobre o gerador de novo código, consulte Geração de código de script.

Os exemplos a seguir mostram o formato do código a ser inserido na caixa de código:

Python

O exemplo a seguir pega o primeiro `DynamicFrame` recebido, converte-o em um `DataFrame` para aplicar o método de filtro nativo (mantendo somente registros que têm mais de 1.000 votos) e, em seguida, converte-o novamente em um `DynamicFrame` antes de devolvê-lo.

```
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

O exemplo a seguir pega o primeiro `DynamicFrame` recebido, converte-o em um `DataFrame` para aplicar o método de filtro nativo (mantendo somente registros que têm mais de 1.000 votos) e, em seguida, converte-o novamente em um `DynamicFrame` antes de devolvê-lo.

```
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)
  }
}
```

Editar o esquema de um nó de transformação personalizada

Quando você usa um nó de transformação personalizada, o AWS Glue Studio não pode inferir automaticamente os esquemas de saída criados pela transformação. Use o editor de esquema para descrever as alterações de esquema implantadas pelo código de transformação personalizada.

Um nó de código personalizado pode ter qualquer número de nós pais, cada um fornecendo um `DynamicFrame` como entrada para o seu código personalizado. Um nó de código personalizado retorna uma coleção de `DynamicFrames`. Cada `DynamicFrame` usado como entrada tem um esquema associado. Você deve adicionar um esquema que descreva cada `DynamicFrame` retornado pelo nó de código personalizado.

> **Note**
>
> Quando você define seu próprio esquema em uma transformação personalizada, o AWS Glue Studio não herda esquemas de nós anteriores. Para atualizar o esquema, selecione o nó Custom transform (Transformação personalizada) e escolha a guia Data preview (Visualização de dados). Depois que a visualização for gerada, escolha “Use Preview Schema” (Usar esquema de visualização). O esquema será substituído pelo esquema usando os dados de visualização.

Para editar os esquemas de um nó de transformação personalizada

1. Com o nó de transformação personalizada selecionado no diagrama de trabalho, no painel de detalhes do nó, escolha a guia Outpu schema (Esquema de saída).
2. Escolha Edit (Editar) para fazer alterações no esquema.

   Se você tiver chaves de propriedade de dados aninhadas, como uma matriz ou objeto, pode escolher o ícone Expand-Rows (Expandir linhas, ) no canto superior direito de cada painel de esquema, para expandir a lista de chaves de propriedade de dados filhas. Depois de escolher esse ícone, ele muda para o ícone Collapse-Rows (Recolher linhas, ), que você pode usar para recolher a lista de chaves de propriedade filhas.
3. Modifique o esquema usando as seguintes ações na seção no lado direito da página:

- Para renomear uma chave de propriedade, coloque o cursor na caixa de texto Key (Chave) da chave de propriedade e insira o novo nome.
- Para alterar o tipo de dados de uma chave de propriedade de dados, use a lista para escolher o novo tipo de dados da chave de propriedade.
- Para adicionar uma nova propriedade de nível superior ao esquema, escolha o botão Overflow (excedente,

)
  à esquerda do botão Cancel (Cancelar) e, em seguida, escolha Add root key (Adicionar chave raiz).
- Para adicionar uma chave de propriedade filha ao esquema, escolha o ícone Add-Key (Adicionar chave,

)
  associado à chave pai. Insira um nome para a chave filha e escolha o tipo de dados.
- Para remover uma chave de propriedade do esquema, escolha o ícone Remove (Excluir,
  
)
  à extrema direita do nome da chave.

4. Se o código de transformação personalizada usar vários `DynamicFrames`, você poderá adicionar esquemas de saída adicionais.

   - Para adicionar um esquema novo e vazio, escolha o ícone Overflow (Excedente,

)
     e, em seguida, escolha Add output schema (Adicionar esquema de saída).
   - Para copiar um esquema existente em um novo esquema de saída, certifique-se de que aquele que você deseja copiar seja exibido no seletor de esquemas. Selecione o ícone Overflow (Excedente,

)
     e, em seguida, escolha Duplicate (Duplicar).

   Se você quiser remover um esquema de saída, certifique-se de que o esquema que deseja copiar seja exibido no seletor de esquemas. Escolha o ícone Overflow (Excedente,

)
   e, em seguida, escolha Delete (Excluir).

5. Adicione novas chaves raiz ao novo esquema ou edite as chaves duplicadas.

6. Quando estiver modificando os esquemas de saída, escolha o botão Apply (Aplicar) para salvar suas alterações e saia do editor de esquemas.

   Se não quiser salvar as alterações, escolha o botão Cancel (Cancelar).

Configurar a saída da transformação personalizada

Uma transformação de código personalizado retorna uma coleção de `DynamicFrames`, mesmo que haja somente um `DynamicFrame` no conjunto de resultados.

Para processar a saída de um nó de transformação personalizada

1. Adicione um nó de transformação SelectFromCollection (Selecionar da coleção) que tem o nó de transformação personalizada como nó pai. Atualize esta transformação para indicar qual conjunto de dados você deseja usar. Consulte Usar SelectFromCollection (Selecionar da coleção) para escolher qual conjunto de dados manter Para mais informações.
2. Adicione mais transformações SelectFromCollection (Selecionar da coleção) para o diagrama de trabalho se você quiser usar `DynamicFrames` adionais produzidos pelo nó de transformação personalizada.

   Considere um cenário no qual você adiciona um nó de transformação personalizada para dividir um conjunto de dados de voo em vários conjuntos de dados, mas duplica algumas das chaves de propriedade de identificação em cada esquema de saída, como a data ou o número do voo. Adicione um nó de transformação SelectFromCollection (Selecionar da coleção) para cada esquema de saída, com o nó de transformação personalizada como nó pai.
3. (Opcional) em seguida, você pode usar cada nó de transformação SelectFromCollection (Selecionar da coleção) como entrada para outros nós no trabalho ou como pai para um nó de destino de dados.

Transformações visuais personalizadas do AWS Glue

As transformações visuais personalizadas permitem que você crie transformações e as disponibilize para uso em trabalhos do AWS Glue Studio. As transformações visuais personalizadas permitem que os desenvolvedores de ETL, que talvez não estejam familiarizados com a codificação, pesquisem e usem uma biblioteca crescente de transformações usando a interface do AWS Glue Studio.

Você pode criar uma transformação visual personalizada e, em seguida, fazer o upload para o Amazon S3 para disponibilizá-la para uso por meio do editor visual no AWS Glue Studio para trabalhar com esses trabalhos.

Tópicos

Conceitos básicos de transformações visuais personalizadas

Para criar uma transformação visual personalizada, siga estas etapas.

- Etapa 1. Crie um arquivo de configuração JSON
- Etapa 2. Implemente a lógica da transformação
- Etapa 3. Valide a transformação visual personalizada
- Etapa 4. Atualize a transformação visual personalizada conforme necessário
- Etapa 5. Use a transformação visual personalizada no AWS Glue Studio

Comece configurando o bucket do Amazon S3 e continue com a Etapa 1. Crie um arquivo de configuração JSON.

Pré-requisitos

As transformações fornecidas pelo cliente residem em uma conta AWS do cliente. Essa conta é proprietária das transformações e, portanto, tem todas as permissões para visualizá-las (pesquisar e usar), editá-las ou excluí-las.

Para usar uma transformação personalizada no AWS Glue Studio, você precisará criar e fazer upload de dois arquivos para o bucket de ativos do Amazon S3 nessa conta AWS:

- Arquivo Python: contém a função de transformação
- Arquivo JSON: descreve a transformação. Isso também é conhecido como o arquivo de configuração que é necessário para definir a transformação.

Para emparelhar os arquivos, use o mesmo nome para ambos. Por exemplo:

- myTransform.json
- myTransform.py

Opcionalmente, você pode dar à sua transformação visual personalizada um ícone personalizado fornecendo um arquivo SVG contendo o ícone. Para emparelhar os arquivos, use o mesmo nome para o ícone:

- myTransform.svg

O AWS Glue Studio os combinará automaticamente usando seus respectivos nomes de arquivo. Os nomes dos arquivos não podem ser os mesmos para nenhum módulo existente.

Convenção recomendada para nome de arquivo de transformação

O AWS Glue Studio importará seu arquivo como módulo (por exemplo, `import myTransform`) em seu script de trabalho. Portanto, o nome do seu arquivo deve seguir as mesmas regras de nomenclatura definidas para nomes de variáveis (identificadores) do Python. Especificamente, eles devem começar com uma letra ou um sublinhado e, em seguida, ser compostos inteiramente por letras, dígitos e/ou sublinhados.

> **Note**
>
> Certifique-se de que o nome do arquivo de transformação não esteja em conflito com os módulos python carregados existentes (por exemplo, `sys, array, copy` etc.) para evitar problemas inesperados de runtime.

Configurar um bucket do Amazon S3

As transformações que você cria são armazenadas no Amazon S3 e pertencem à sua conta AWS. Você cria novas transformações visuais personalizadas simplesmente carregando arquivos (json e py) para a pasta de ativos do Amazon S3, onde todos os scripts de trabalho estão armazenados atualmente (por exemplo, `s3://aws-glue-assets-<accountid>-<region>/transforms`). Se estiver usando um ícone personalizado, carregue-o também. Por padrão, o AWS Glue Studio lerá todos os arquivos .json da pasta /transforms no mesmo bucket do S3.

Etapa 1. Crie um arquivo de configuração JSON

É necessário um arquivo de configuração JSON para definir e descrever sua transformação visual personalizada. O esquema para o arquivo de configuração é como se segue.

Estrutura de arquivo JSON

Campos

- `name: string`: (obrigatório) o nome do sistema de transformação usado para identificar as transformações. Siga as mesmas regras de nomenclatura definidas para nomes de variáveis (identificadores) do Python. Especificamente, eles devem começar com uma letra ou um sublinhado e, em seguida, ser compostos inteiramente por letras, dígitos e/ou sublinhados.
- `displayName: string`: (opcional) o nome da transformação exibida no editor visual de trabalhos do AWS Glue Studio. Se não for especificado nenhum `displayName`, o `name` será usado como o nome da transformação no AWS Glue Studio.
- `description: string`: (opcional) a descrição da transformação é exibida no AWS Glue Studio e pode ser pesquisada.
- `functionName: string`: (obrigatório) o nome da função do Python é usado para identificar a função a ser chamada no script do Python.
- `path: string`: (opcional) o caminho completo do Amazon S3 para o arquivo fonte do Python. Se não for especificado, o AWS Glue usará a correspondência de nomes de arquivo para emparelhar os arquivos .json e .py. Por exemplo, o nome do arquivo JSON, `myTransform.json`, será emparelhado com o arquivo Python, `myTransform.py`, no mesmo local do Amazon S3.
- `parameters: Array of TransformParameter object`: (opcional) a lista de parâmetros a serem exibidos quando você os configura no editor visual do AWS Glue Studio.

Campos TransformParameter

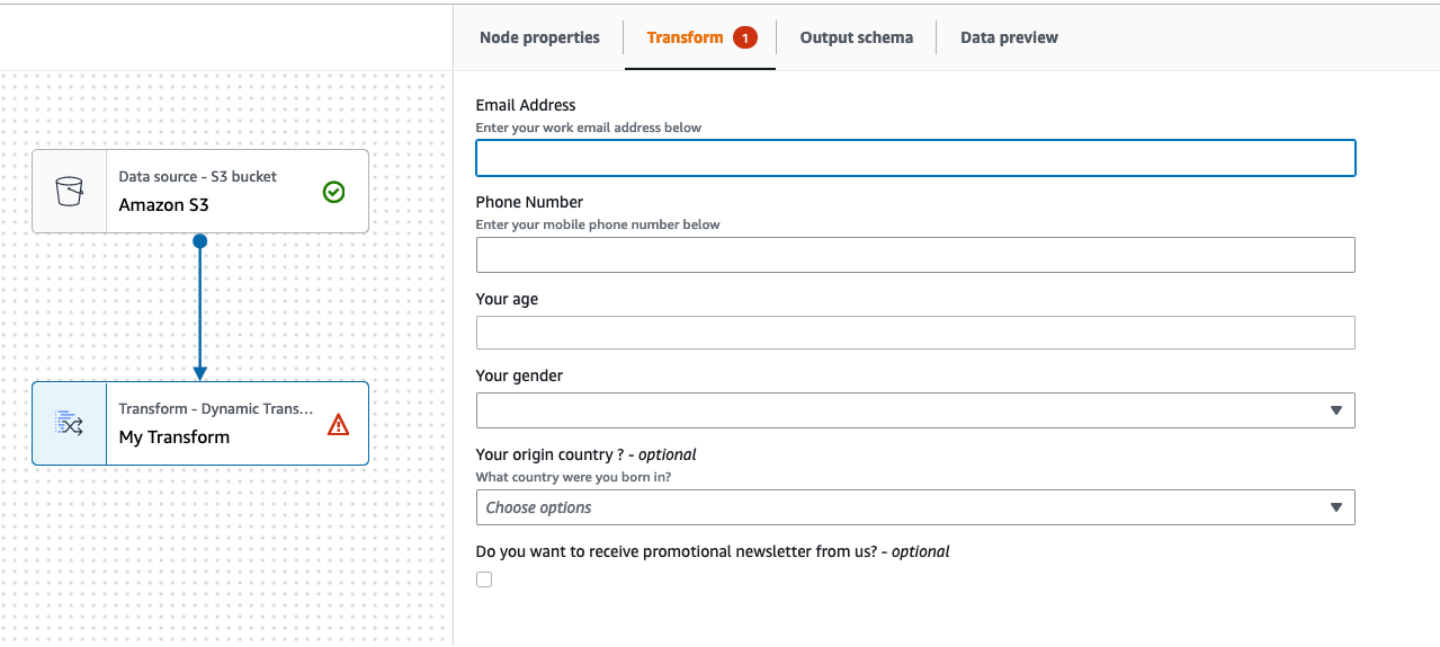
- `name: string`: (obrigatório) o nome do parâmetro que será passado para a função do Python como um argumento nomeado no script do trabalho. Siga as mesmas regras de nomenclatura definidas para nomes de variáveis (identificadores) do Python. Especificamente, eles devem começar com uma letra ou um sublinhado e, em seguida, ser compostos inteiramente por letras, dígitos e/ou sublinhados.
- `displayName: string`: (opcional) o nome da transformação exibida no editor visual de trabalhos do AWS Glue Studio. Se não for especificado nenhum `displayName`, o `name` será usado como o nome da transformação no AWS Glue Studio.
- `type: string`: (obrigatório) o tipo de parâmetro que aceita tipos de dados comuns do Python. Valores válidos: 'str' | 'int' | 'float' | 'list' | 'bool'.
- `isOptional: boolean`: (opcional) determina se o parâmetro é opcional. Por padrão, todos os parâmetros são obrigatórios.
- `description: string`: (opcional) a descrição é exibida no AWS Glue Studio para ajudar o usuário a configurar o parâmetro de transformação.
- `validationType: string`: (opcional) define a forma como esse parâmetro é validado. Atualmente, só aceita expressões regulares. Por padrão, o tipo de validação é definido como `RegularExpression`.
- `validationRule: string`: (opcional) expressão regular usada para validar a entrada do formulário antes do envio, quando `validationType` está definido como `RegularExpression`. A sintaxe da expressão regular deve ser compatível com as especificações RegExp do Ecmascript.
- `validationMessage: string`: (opcional) a mensagem a ser exibida quando a validação falhar.
- `listOptions: An array of TransformParameterListOption object` OU um `string` ou o valor da string 'coluna': (opcional) opções a serem exibidas no controle de interface de usuário de seleção ou seleção múltipla. Aceitar uma lista de valores separados por vírgula ou um objeto JSON do tipo `TransformParameterListOption`. Ele também pode preencher dinamicamente a lista de colunas do esquema do nó pai especificando o valor da string “coluna”.
- `listType: string`: (opcional) defina os tipos de opções como tipo = “list”. Valores válidos: 'str' | 'int' | 'float' | 'list' | 'bool'. Tipo de parâmetro que aceita tipos de dados comuns do Python.

Campos TransformParameterListOption

- `value: string | int | float | bool`: (obrigatório) valor da opção.
- `label: string`: (opcional) rótulo da opção exibida na lista suspensa de seleção.

Transformar parâmetros em AWS Glue Studio

Por padrão, os parâmetros são obrigatórios, a menos que sejam marcados como `isOptional` no arquivo .json. No AWS Glue Studio, os parâmetros são exibidos na guia Transform (Transformar). O exemplo mostra parâmetros definidos pelo usuário, como endereço de e-mail, número de telefone, idade, sexo e país de origem.

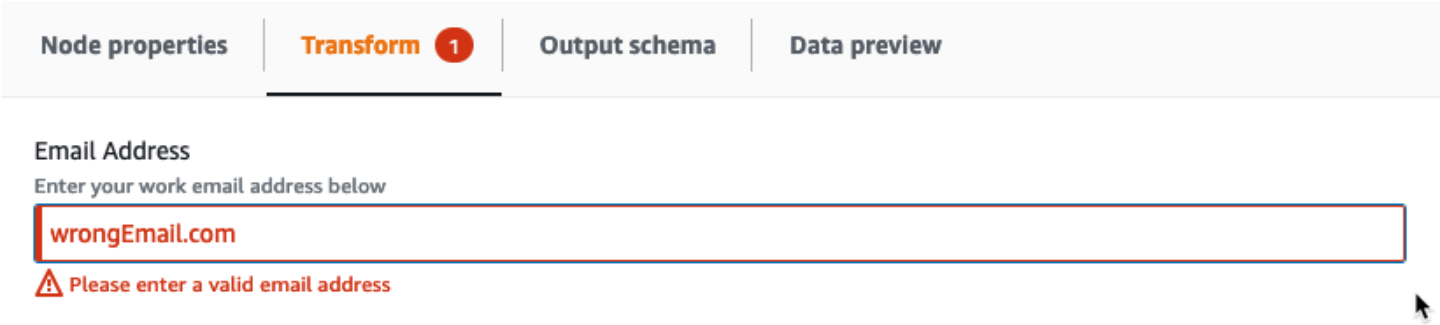


Você pode impor algumas validações no AWS Glue Studio usando expressões regulares no arquivo json especificando o parâmetro `validationRule` e especificando uma mensagem de validação em `validationMessage`.

```
    "validationRule": "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$",
    "validationMessage": "Please enter a valid US number"
```

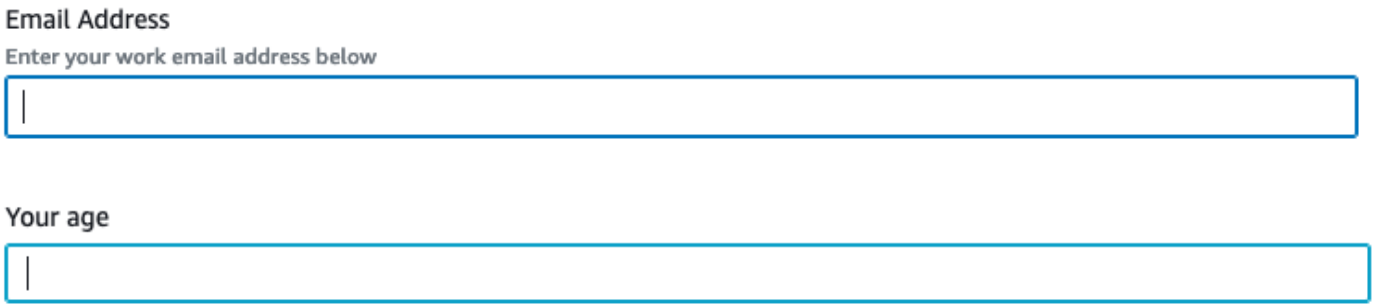
> **Note**
>
> Como a validação ocorre no navegador, a sintaxe da expressão regular deve ser compatível com as especificações RegExp do Ecmascript. A sintaxe do Python não é compatível com essas expressões regulares.

Adicionar validação evitará que o usuário salve o trabalho com a entrada incorreta do usuário. O AWS Glue Studio exibe a mensagem de validação como a exibida no exemplo:

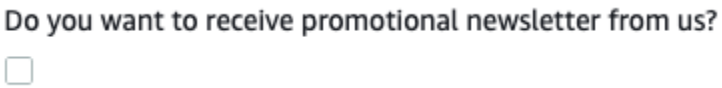


Os parâmetros são exibidos no AWS Glue Studio com base na configuração do parâmetro.

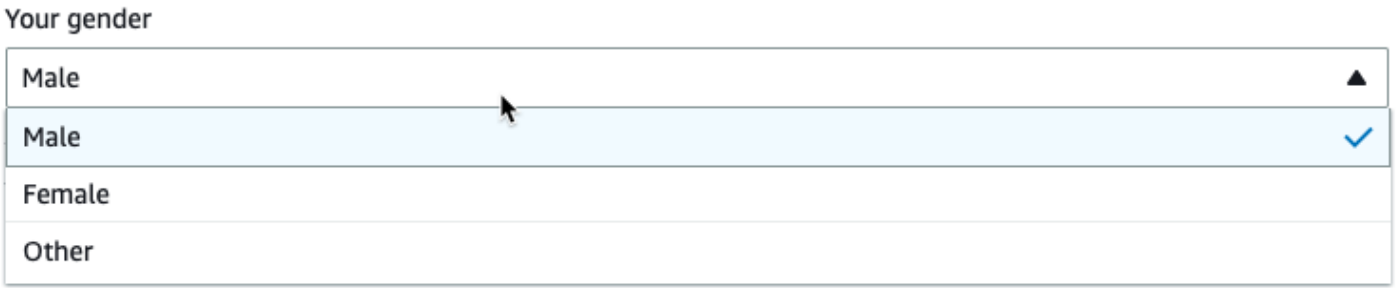
- Quando o `type` é qualquer um dos seguintes: `str`, `int` ou `float`, um campo de entrada de texto é exibido. Por exemplo, a exibição de tela mostra campos de entrada para os parâmetros 'Endereço de e-mail' e 'Sua idade'.



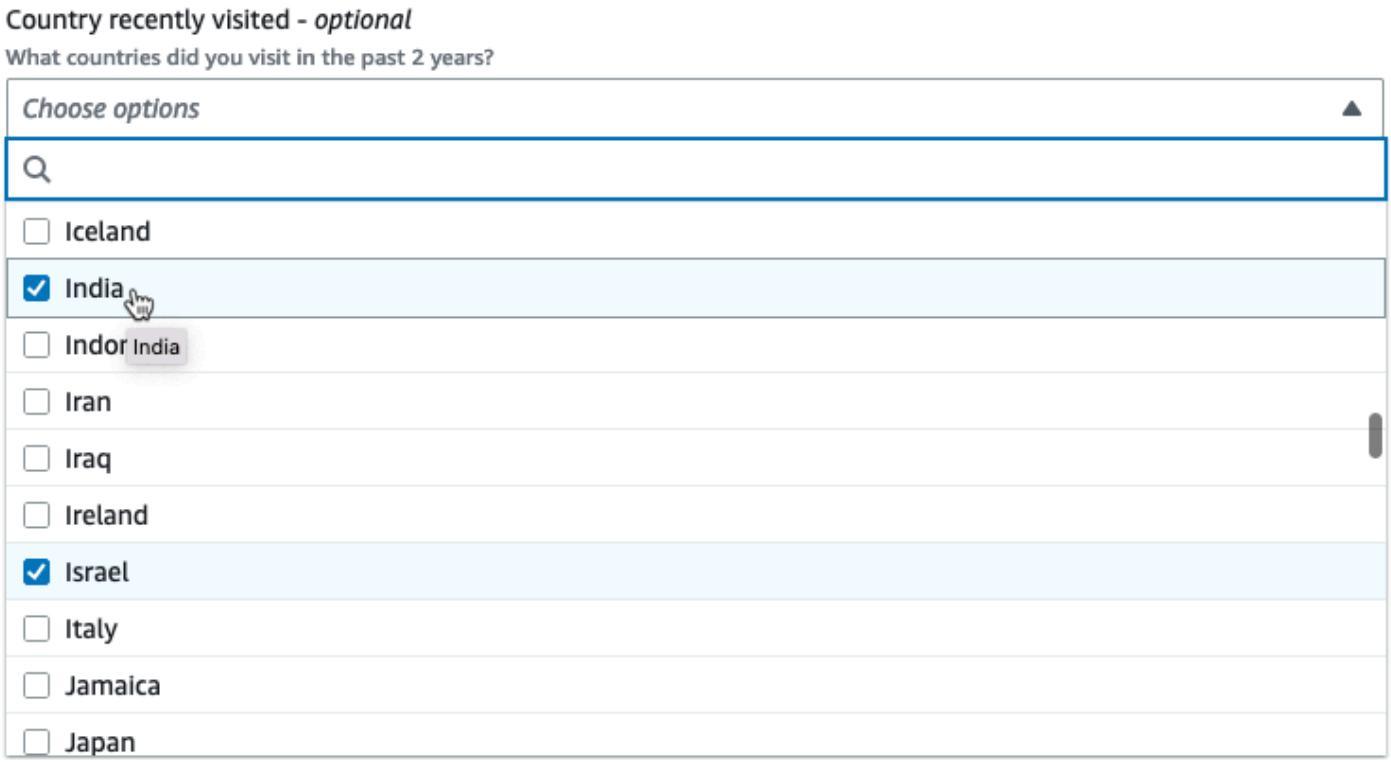
- Quando `type` é `bool`, uma caixa de seleção é exibida.



- Quando `type` é `str` e `listOptions` é fornecida, é exibida uma única lista de seleção.



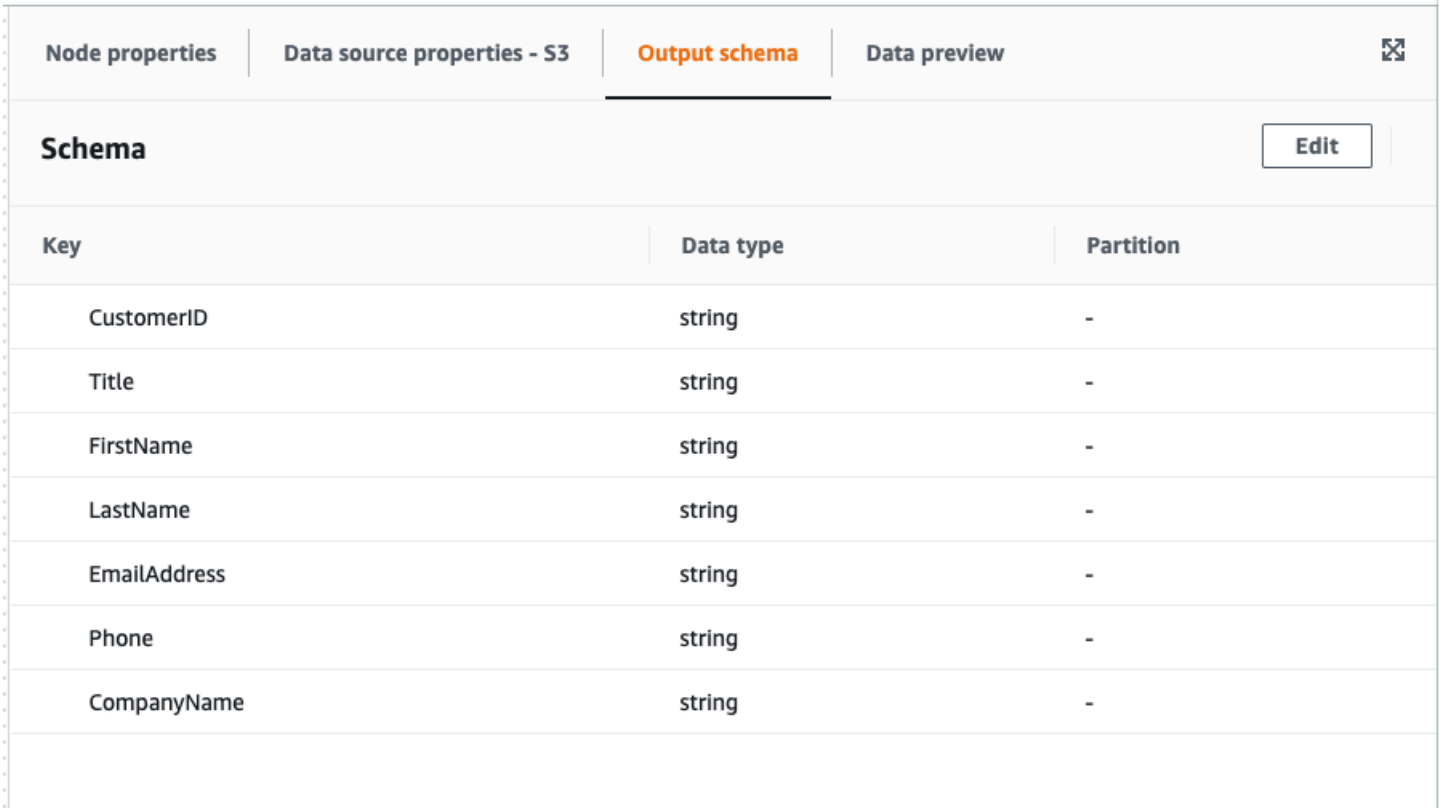
- Quando são fornecidos `type`, `list`, `listOptions` e `listType`, é exibida uma lista de seleção múltipla.



Exibir um seletor de coluna como parâmetro

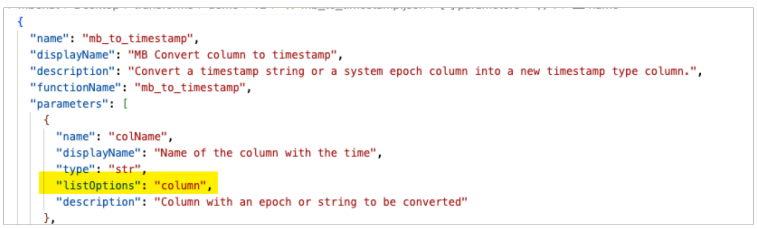
Se a configuração exigir que o usuário escolha uma coluna do esquema, você poderá exibir um seletor de coluna para que o usuário não precise digitar o nome da coluna. Ao definir o campo `listOptions` como “coluna”, o AWS Glue Studio exibe dinamicamente um seletor de coluna com base no esquema de saída do nó pai. O AWS Glue Studio pode exibir um seletor de coluna única ou múltipla.

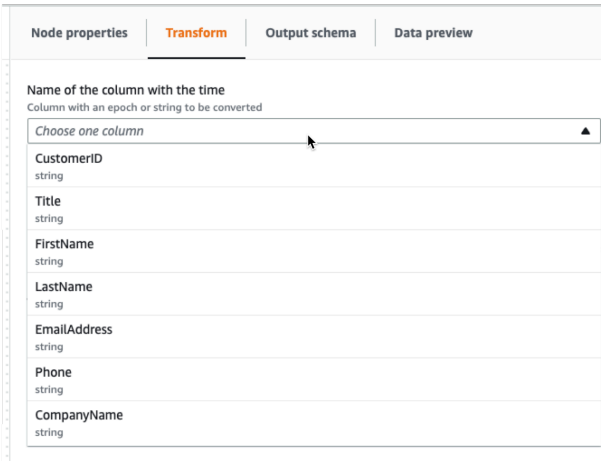
O exemplo a seguir usa o esquema:



Para definir seu parâmetro Transformação visual personalizada para exibir uma única coluna:

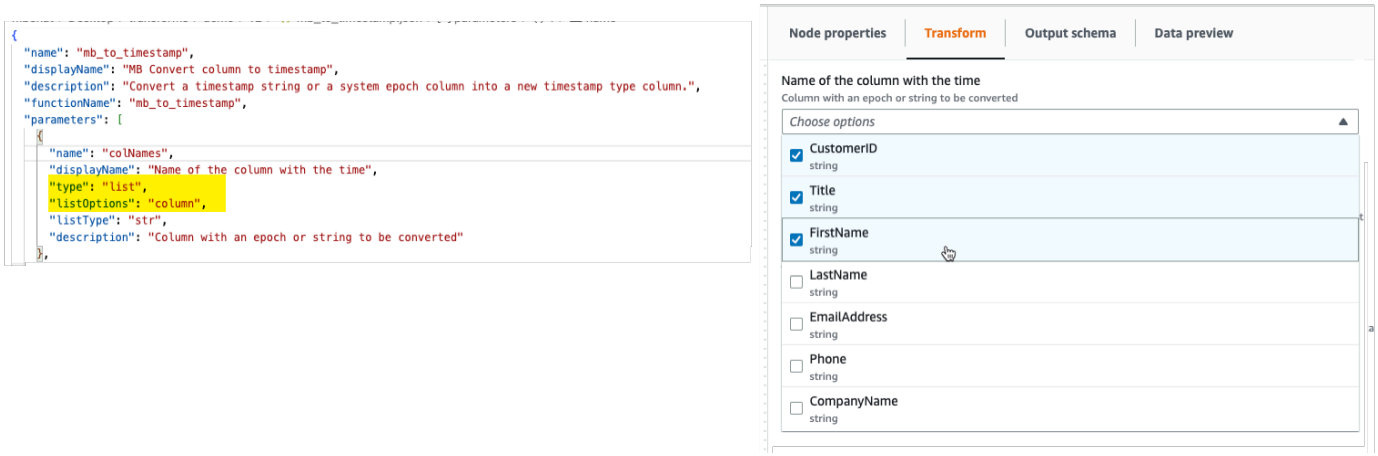
1. No seu arquivo JSON, para o objeto `parameters`, defina o valor de `listOptions` como "coluna". Isso permite que o usuário escolha uma coluna em uma lista de seleção no AWS Glue Studio.





2. Você também pode permitir a seleção de várias colunas definindo o parâmetro como:

   - `listOptions: "column"`
   - `type: "list"`



Etapa 2. Implemente a lógica da transformação

> **Note**
>
> As transformações visuais personalizadas são compatíveis apenas scripts Python. O Scala não é compatível.

Para adicionar o código que implementa a função definida pelo arquivo de configuração .json, é recomendável colocar o arquivo Python no mesmo local do arquivo .json, com o mesmo nome, mas com a extensão ".py". O AWS Glue Studioemparelha automaticamente os arquivos .json e .py para que você não precise especificar o caminho do arquivo Python no arquivo de configuração.

No arquivo Python, adicione a função declarada, com os parâmetros nomeados configurados e registre-a para ser usada no `DynamicFrame`. Este é um exemplo de um arquivo 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
```

É recomendável usar um caderno AWS Glue para obter a maneira mais rápida de desenvolver e testar o código python. Consulte Conceitos básicos de cadernos no AWS Glue Studio.

Para ilustrar como implementar a lógica de transformação, a transformação visual personalizada no exemplo abaixo é uma transformação para filtrar os dados recebidos para manter somente os dados relacionados a um estado específico dos EUA. O arquivo .json contém o parâmetro para `functionName` como `custom_filter_state` e dois argumentos (“state” e “colName” com o tipo “str”).

O exemplo de arquivo config .json é:

```
{
"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"
   }
  ]
}
```

Para implementar o script complementar no Python

1. Inicie um caderno do AWS Glue e execute a célula inicial fornecida para o início da sessão. A execução da célula inicial cria os componentes básicos necessários.
2. Crie uma função que execute a filtragem conforme descrito no exemplo e registre-a no `DynamicFrame`. Copie o código abaixo e cole em uma célula do caderno do 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. Crie ou carregue dados de amostra para testar o código na mesma célula ou em uma nova célula. Se você adicionar os dados de exemplo em uma nova célula, não se esqueça de executar a célula. Por exemplo:

```
# 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. Teste para validar o "custom_filter_state" com argumentos diferentes:

```
[14]: dynf1.custom_filter_state("state", "NY").show()
      {"count": 2, "state": "NY"}
```

5. Depois de executar vários testes, salve o código com a extensão .py e nomeie o arquivo .py com um nome que espelhe o nome do arquivo .json. Os arquivos.py e .json devem estar na mesma pasta de transformação.

   Copie o código a seguir e cole-o em um arquivo e renomeie-o com uma extensão de arquivo .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. No AWS Glue Studio, abra uma tarefa visual e adicione a transformação à tarefa selecionando-a na lista de Transformações disponíveis.

   Para reutilizar essa transformação em um código de script Python, adicione o caminho do Amazon S3 ao arquivo .py nno trabalho, em “Referenced files path” (Caminho de arquivos referenciados) e, no script, importe o nome do arquivo python (sem a extensão) adicionando-o à parte superior do arquivo. Por exemplo: `import` <nome do arquivo (sem a extensão)>

Etapa 3. Validar e solucionar problemas de transformações visuais personalizadas no AWS Glue Studio

O AWS Glue Studio valida o arquivo de configuração JSON antes que as transformações visuais personalizadas sejam carregadas no AWS Glue Studio. A validação inclui:

- Presença de campos obrigatórios
- Validação do formato JSON
- Parâmetros incorretos ou inválidos
- Presença dos arquivos .py e .json no mesmo caminho do Amazon S3
- Nomes de arquivo correspondentes para o.py e .json

Se a validação for bem-sucedida, a transformação será listada na lista de ações disponíveis no editor visual. Se um ícone personalizado tiver sido fornecido, ele deve estar visível ao lado da Ação.

Se a validação falhar, o AWS Glue Studio não carrega a transformação visual personalizada.

Etapa 4. Atualize transformações visuais personalizadas conforme necessário

Depois de criado e usado, o script de transformação pode ser atualizado desde que a transformação siga a definição json correspondente:

- O nome usado ao atribuir ao DynamicFrame deve corresponder ao `functionName` json.
- Os argumentos da função devem ser definidos no arquivo json conforme descrito em Etapa 1. Crie um arquivo de configuração JSON .
- O caminho do arquivo Python no Amazon S3 não pode mudar, pois os trabalhos dependem diretamente dele.

> **Note**
>
> Se alguma atualização precisar ser feita, certifique-se de que o script e o arquivo .json sejam atualizados de forma consistente e que todas as tarefas visuais sejam salvas corretamente novamente com a nova transformação. Se os trabalhos visuais não forem salvos depois que as atualizações forem feitas, as atualizações não serão aplicadas e validadas. Se o arquivo de script Python for renomeado ou não for colocado ao lado do arquivo .json, será necessário especificar o caminho completo no arquivo .json.

Ícone personalizado

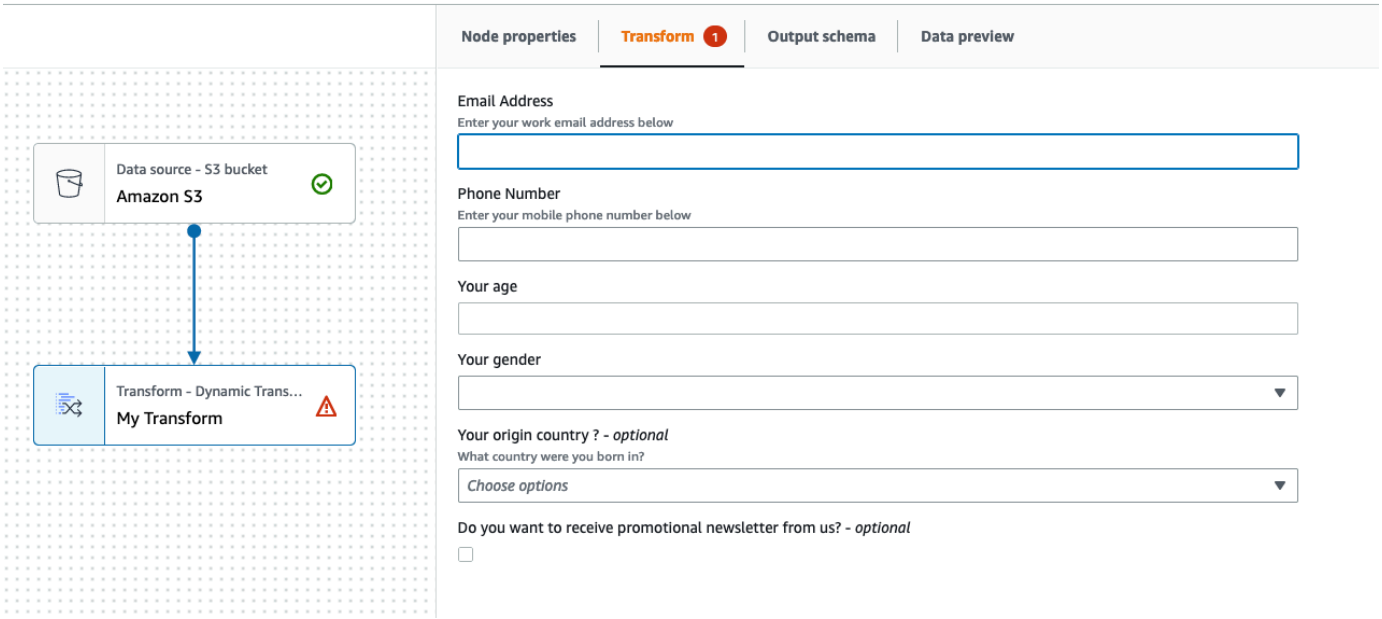
Se você determinar que o ícone padrão de sua Ação não a distingue visualmente como parte de seus fluxos de trabalho, você poderá fornecer um ícone personalizado, conforme descrito em the section called “ Conceitos básicos de transformações visuais personalizadas ”. Você pode atualizar o ícone atualizando o SVG correspondente hospedado no Amazon S3.

Para obter melhores resultados, crie sua imagem para ser visualizada em 32x32px seguindo as diretrizes do Cloudscape Design System. Para obter mais informações sobre o uso do CloudWatch, consulte a documentação do Cloudscape

Etapa 5. Use a transformações visuais personalizadas no AWS Glue Studio

Para usar uma transformação visual personalizada no AWS Glue Studio, você carrega os arquivos de configuração e de origem e seleciona a transformação no menu Action (Ação). Todos os parâmetros que precisem de valores ou entradas estão disponíveis para você na guia Transform (Transformar).

1. Faça o upload dos dois arquivos (arquivo fonte do Python e arquivo de configuração JSON) para a pasta de ativos do Amazon S3, onde os scripts de trabalho são armazenados. Por padrão, o AWS Glue lerá todos os arquivos .json da pasta /transforms no mesmo bucket do Amazon S3.
2. No menu Action (Ação), escolha a transformação visual personalizada. Ela é nomeada com a transformação `displayName` ou o nome que você especificou no arquivo de configuração .json.
3. Insira valores para os parâmetros que foram configurados no arquivo de configuração.



Exemplos de uso

A seguir está um exemplo de todos os parâmetros possíveis em um arquivo de configuração .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
    }
  ]
}
```

Exemplos de scripts visuais personalizados

Os exemplos a seguir realizam transformações equivalentes. No entanto, o segundo exemplo (SparkSQL) é o mais limpo e eficiente, seguido pela UDF do pandas e, finalmente, pelo mapeamento de baixo nível no primeiro exemplo. O exemplo a seguir é um exemplo completo de uma transformação simples para somar duas colunas:

```
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
```

O exemplo a seguir é uma transformação equivalente usando a API do 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
```

O exemplo a seguir usa as mesmas transformações, mas usando uma UDF do pandas, que é mais eficiente do que usar uma UDF simples. Para obter mais informações sobre como escrever UDFs panda, consulte a documentação do 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

O vídeo a seguir fornece uma introdução às transformações visuais personalizadas e demonstra como usá-las.

Usar estruturas do Data Lake com o AWS Glue Studio

Visão geral

As estruturas de data lake de código aberto simplificam o processamento incremental de dados para os arquivos armazenados em data lakes criados no Amazon S3. O AWS Glue 3.0 e posteriores são compatíveis com as seguintes estruturas de armazenamento em data lake de código aberto:

- Apache Hudi
- Linux Foundation Delta Lake
- Apache Iceberg

A partir do AWS Glue 4.0, o AWS Glue fornece suporte nativo para essas estruturas para que você possa ler e gravar os dados que armazenar no Amazon S3 de maneira consistente em termos de transações. Não é necessário instalar um conector separado nem realizar etapas adicionais de configuração para usar essas estruturas em trabalhos do AWS Glue.

As estruturas do Data Lake podem ser usadas como fonte ou destino no AWS Glue Studio por meio de trabalhos do editor de scripts do Spark. Para obter mais informações sobre o uso do Apache Hudi, do Apache Iceberg e do Delta Lake, consulte: Usar estruturas de data lake com trabalhos do AWS Glue ETL.

Criação de formatos de tabela abertos a partir de uma fonte de streaming do AWS Glue

As tarefas de ETL de streaming do AWS Glue consomem continuamente dados de fontes de streaming, limpam e transformam os dados em andamento e os disponibilizam para análise em questão de segundos.

A AWS oferece uma ampla seleção de serviços para atender às suas necessidades. Um serviço de replicação de banco de dados, como o AWS Database Migration Service, pode replicar os dados de seus sistemas de origem para o Amazon S3, que normalmente hospeda a camada de armazenamento do data lake. Embora seja fácil aplicar atualizações em um sistema de gerenciamento de banco de dados relacional (RDBMS) que oferece suporte a uma aplicação de origem online, é difícil aplicar esse processo de CDC em seus data lakes. As estruturas de gerenciamento de dados de código aberto simplificam o processamento incremental de dados e o desenvolvimento de pipelines de dados e são uma boa opção para resolver esse problema.

Para obter mais informações, consulte:

- Criar um data lake transacional quase em tempo real baseado em Apache HUDI usando streaming do AWS Glue
- Crie um data lake Apache Iceberg em tempo real alinhado ao GDPR

Usar a estrutura do Hudi no AWS Glue Studio

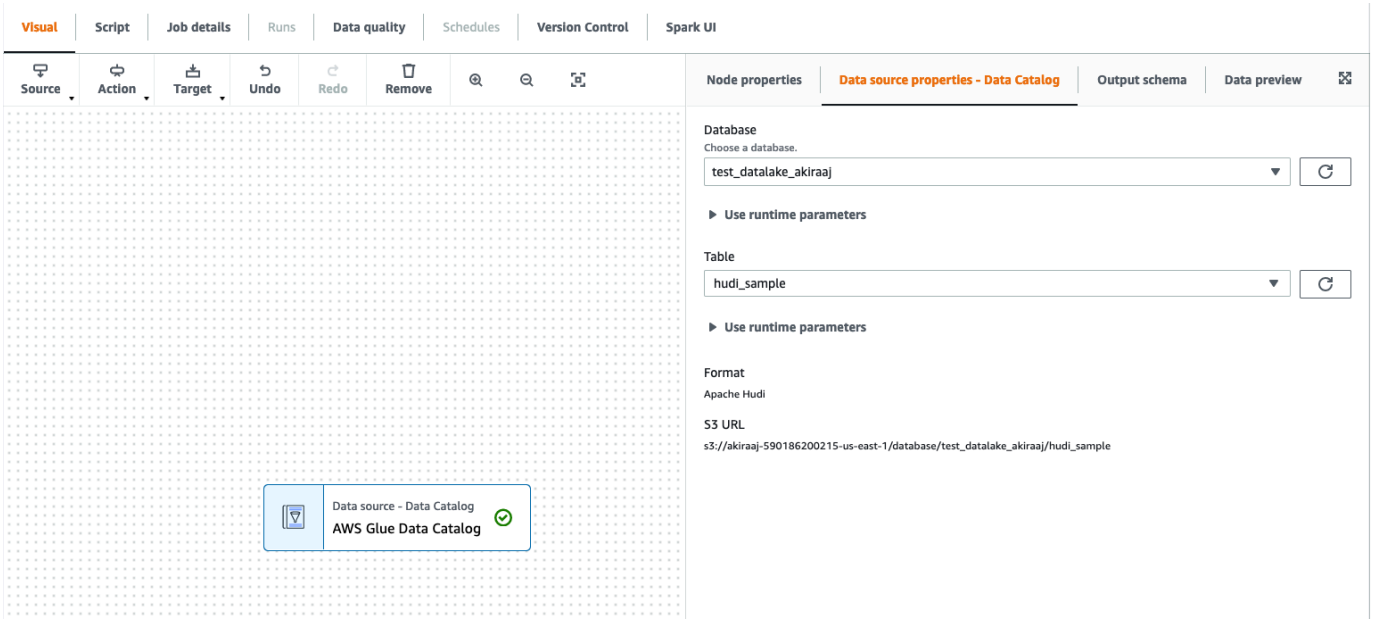
Ao criar ou editar um trabalho, o AWS Glue Studio adiciona automaticamente as bibliotecas do Hudi correspondentes, dependendo da versão do AWS Glue que você esteja usando. Para obter mais informações, consulte Usar a estrutura Hudi no AWS Glue.

Usar a estrutura do Apache Hudi em fontes de dados do Data Catalog

Para adicionar um formato de fonte de dados Hudi a um trabalho:

1. No menu Fonte, escolha AWS Glue Studio Data Catalog.

2. Na guia Propriedades da fonte de dados, escolha um banco de dados e uma tabela.
3. O AWS Glue Studio exibe o tipo de formato como Apache Hudi e o URL do Amazon S3.



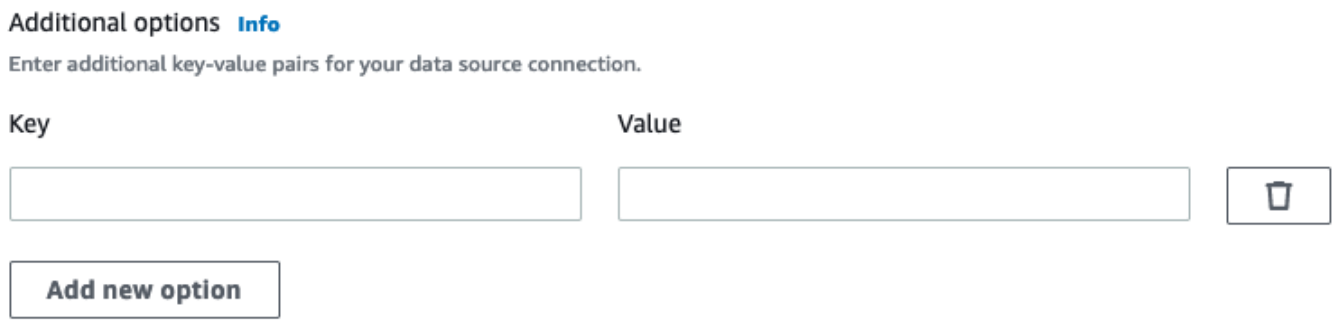
Usar a estrutura do Hudi em fontes de dados do Amazon S3

1. No menu Fonte, selecione Amazon S3.
2. Se você escolher a tabela do Data Catalog como o tipo de fonte do Amazon S3, escolha um banco de dados e uma tabela.
3. O AWS Glue Studio exibe o formato como Apache Hudi e o URL do Amazon S3.
4. Se você escolher o local do Amazon S3 como o tipo de fonte do Amazon S3, escolha a URL do S3 clicando em Procurar o Amazon S3.
5. Em Formato de dados, selecione Apache Hudi.

> **Note**
>
> Se o AWS Glue Studio não conseguir inferir o esquema da pasta ou arquivo do Amazon S3 selecionado, escolha Opções adicionais para selecionar uma nova pasta ou arquivo. Em Opções adicionais, escolha entre as seguintes opções em Inferência de esquema:
>
> - Deixe o AWS Glue Studio escolher automaticamente um arquivo de amostra. O AWS Glue Studio escolherá um arquivo de amostra no local do Amazon S3 para que o esquema possa ser inferido. No campo Arquivo amostrado automaticamente, você pode visualizar o arquivo que foi selecionado automaticamente.

- Escolha um arquivo de amostra do Amazon S3. Escolha o arquivo do Amazon S3 a ser usado clicando em Procurar no Amazon S3.

6. Clique em Inferir esquema. Depois, você poderá visualizar o esquema de saída clicando na guia Esquema de saída.
7. Escolha Opções adicionais para inserir um par de chave-valor.



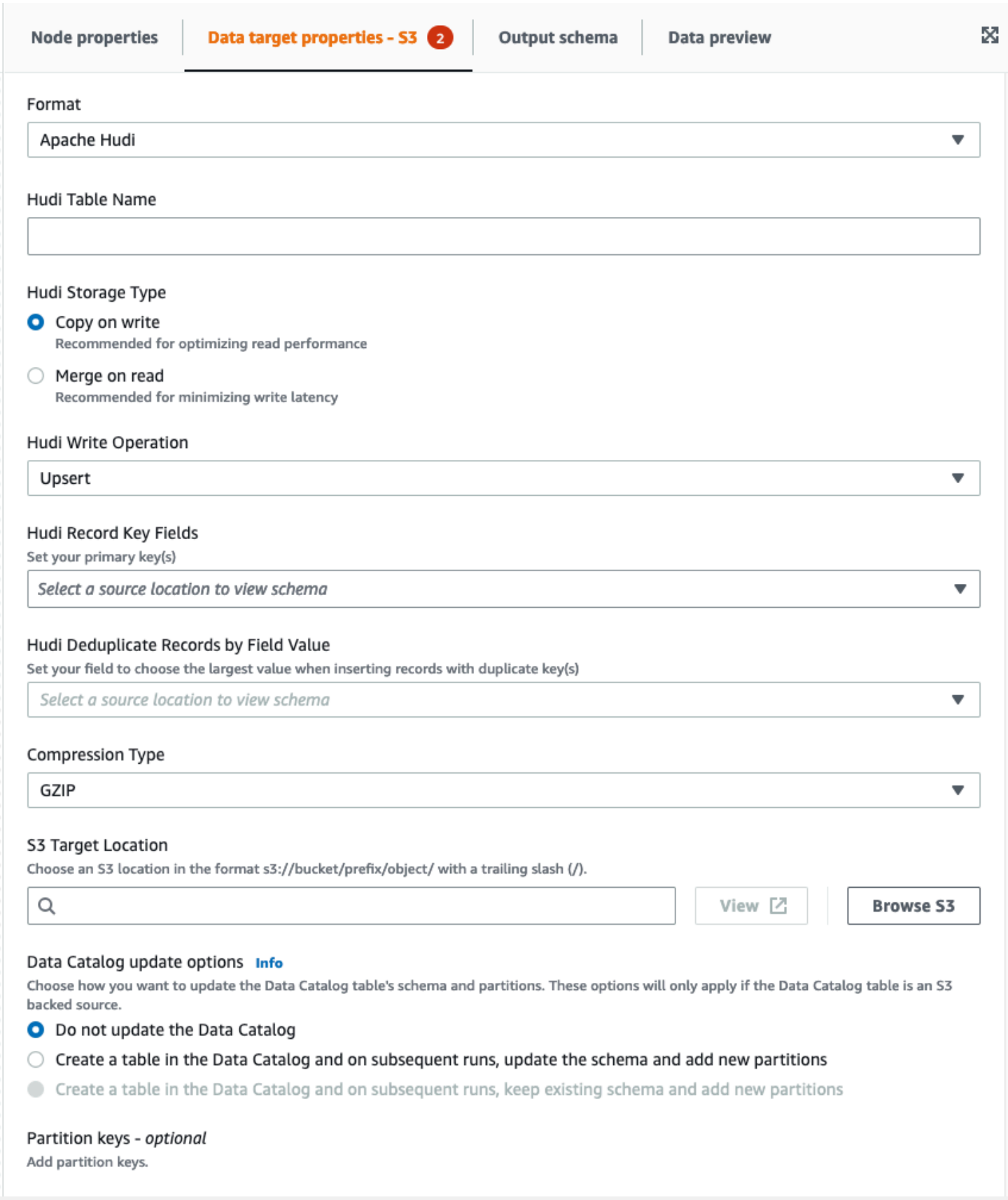
Usar a estrutura do Apache Hudi em destinos de dados

Usar a estrutura do Apache Hudi em destinos de dados do Data Catalog

1. No menu Destino, escolha AWS Glue Studio Data Catalog.
2. Na guia Propriedades da fonte de dados, escolha um banco de dados e uma tabela.
3. O AWS Glue Studio exibe o tipo de formato como Apache Hudi e o URL do Amazon S3.

Usar a estrutura do Apache Hudi em destinos de dados do Amazon S3

Insira os valores ou selecione-os entre as opções disponíveis para configurar o formato do Apache Hudi. Para obter mais informações sobre o Apache Hudi, consulte a documentação do 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.

- Nome da tabela do Hudi: esse é o nome da tabela do Hudi.
- Tipo de armazenamento do Hudi: escolha entre duas opções:
  - Copiar e gravar: recomendado para otimizar o desempenho de leitura. Esse é o tipo de armazenamento padrão do Hudi. Cada atualização cria uma nova versão dos arquivos durante uma gravação.
  - Mesclar na leitura: recomendado para minimizar a latência de gravação. As atualizações são registradas em arquivos delta baseados em linha e são compactadas conforme necessário para criar novas versões dos arquivos colunares.
- Operação de gravação do Hudi: escolha uma das seguintes opções:
  - Upsert: essa é a operação padrão, na qual os registros de entrada são marcados primeiro como inserções ou atualizações consultando o índice. Recomendado quando você está atualizando dados existentes.
  - Inserir: essa opção insere registros, mas não verifica os registros existentes e pode resultar em duplicatas.
  - Inserção em massa: essa opção insere registros e é recomendado para grandes volumes de dados.
- Campos de chave de registro do Hudi: use a barra de pesquisa para pesquisar e escolher chaves de registro primárias. Os registros no Hudi são identificados por uma chave primária, que é um par de chave de registro e o caminho da partição ao qual o registro pertence.
- Campo de combinação prévia do Hudi: esse é o campo usado para a combinação prévia antes da gravação real. Quando dois registros tiverem o mesmo valor de chave, o AWS Glue Studio selecionará o de maior valor para o campo de combinação prévia. Defina um campo com o valor incremental (por exemplo, updated_at) pertence a.
- Tipo de compactação: escolha uma das opções de tipo de compactação: não compactado, GZIP, LZO ou Snappy.
- Local de destino do Amazon S3: escolha o local de destino do Amazon S3 clicando em Procurar no S3.
- Opções de atualização do Data Catalog: escolha uma das seguintes opções:
  - Do not update the Data Catalog (Não atualizar o Data Log): (padrão) escolha essa opção se você não quiser que o trabalho atualize o Data Catalog, mesmo que o esquema seja alterado ou novas partições sejam adicionadas.
  - Criar uma tabela no Data Catalog e em execuções subsequentes, atualizar o esquema e adicionar novas partições: se você escolher essa opção, o trabalho criará a tabela no Data Catalog quando for executado pela primeira vez. Em execuções de trabalho subsequentes,

ele atualizará a tabela do Data Catalog, se o esquema for alterado ou novas partições forem adicionadas.

Você também deve selecionar um banco de dados no Data Catalog e inserir um nome de tabela.

- Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions (Criar uma tabela no Data Catalog e em execuções subsequentes, manter o esquema atual e adicionar novas partições): se você escolher essa opção, o trabalho criará a tabela no Data Catalog na sua primeira execução. Em execuções de trabalho subsequentes, ele atualizará a tabela do Data Catalog, mas apenas para adicionar novas partições.

  Você também deve selecionar um banco de dados no Data Catalog e inserir um nome de tabela.

- Partition keys (Chaves de partição): escolha quais colunas usar como chaves de particionamento na saída. Para adicionar mais chaves de partição, escolha Add a partition key (Adicionar uma chave de partição.
- Opções adicionais: insira um par de valor-chave conforme necessário.

Gerar código por meio do AWS Glue Studio

Quando o trabalho é salvo, os seguintes parâmetros de trabalho são adicionados a ele se uma fonte ou um destino Hudi forem detectados:

- `--datalake-formats`: uma lista distinta de formatos de data lake detectados no trabalho visual (diretamente escolhendo um “formato” ou indiretamente selecionando uma tabela de catálogo com suporte de um data lake).
- `--conf` : gerado com base no valor de `--datalake-formats`. Por exemplo, se o valor de `--datalake-formats` for "hudi", o AWS Glue gerará um valor `spark.serializer=org.apache.spark.serializer.KryoSerializer –conf spark.sql.hive.convertMetastoreParquet=false` para esse parâmetro.

Substituir as bibliotecas fornecidas pelo AWS Glue

Para usar uma versão do Hudi que não seja compatível com o AWS Glue, você pode especificar seus próprios arquivos JAR de biblioteca do Hudi. Para usar seu próprio arquivo JAR:

- use o parâmetro de trabalho `--extra-jars`. Por exemplo, .`'--extra-jars': 's3pathtojarfile.jar'` Para obter mais informações, consulte parâmetros de trabalho do AWS Glue.

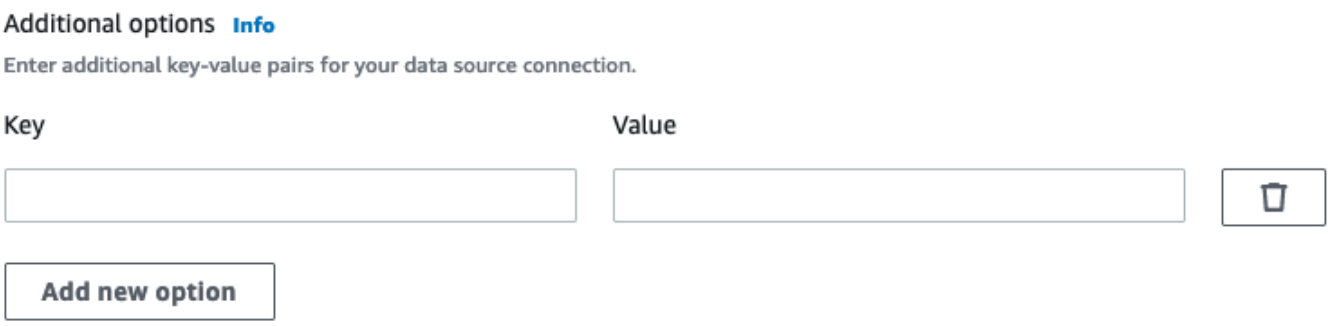
- não inclua `hudi` como um valor para o parâmetro de trabalho `--datalake-formats`. Inserir uma string em branco como um valor garante que nenhuma biblioteca de data lake seja fornecida a você automaticamente pelo AWS Glue. Para obter mais informações, consulte Usar a estrutura Hudi no AWS Glue.

Usar a estrutura do Delta Lake no AWS Glue Studio Glue

Usar a estrutura do Delta Lake em fontes de dados

Usar a estrutura do Delta Lake em fontes de dados do Amazon S3

1. No menu Fonte, selecione Amazon S3.
2. Se você escolher a tabela do Data Catalog como o tipo de fonte do Amazon S3, escolha um banco de dados e uma tabela.
3. O AWS Glue Studio exibe o formato como Data Lake e o URL do Amazon S3.
4. Escolha Opções adicionais para inserir um par de chave-valor. Por exemplo, um par de chave-valor pode ser: chave: timestampAsOf e valor: 2023-02-24 14:16:18.



5. Se você escolher o local do Amazon S3 como o tipo de fonte do Amazon S3, escolha a URL do S3 clicando em Procurar o Amazon S3.
6. Em Formato de dados, escolha Delta Lake.

> **Note**
>
> Se o AWS Glue Studio não conseguir inferir o esquema da pasta ou arquivo do Amazon S3 selecionado, escolha Opções adicionais para selecionar uma nova pasta ou arquivo. Em Opções adicionais, escolha entre as seguintes opções em Inferência de esquema:

- Deixe o AWS Glue Studio escolher automaticamente um arquivo de amostra. O AWS Glue Studio escolherá um arquivo de amostra no local do Amazon S3 para que o esquema possa ser inferido. No campo Arquivo amostrado automaticamente, você pode visualizar o arquivo que foi selecionado automaticamente.
- Escolha um arquivo de amostra do Amazon S3. Escolha o arquivo do Amazon S3 a ser usado clicando em Procurar no Amazon S3.

7. Clique em Inferir esquema. Depois, você poderá visualizar o esquema de saída clicando na guia Esquema de saída.

Usar a estrutura do Delta Lake em fontes de dados do Data Catalog

1. No menu Fonte, escolha AWS Glue Studio Data Catalog.
2. Na guia Propriedades da fonte de dados, escolha um banco de dados e uma tabela.
3. O AWS Glue Studio exibe o tipo de formato como Data Lake e o URL do Amazon S3.

> **Note**
>
> Se sua fonte Delta Lake ainda não estiver registrada na tabela do AWS Glue Data Catalog, você tem duas opções:
>
> 1. Crie um crawler do AWS Glue para o armazenamento de dados Delta Lake. Para obter mais informações, consulte Como especificar opções de configuração para um armazenamento de dados do Delta Lake.
> 2. Use uma fonte de dados do Amazon S3 para selecionar sua fonte de dados Delta Lake. Consulte Usar a estrutura do Delta Lake em fontes de dados do Amazon S3 .

Usar formatos do Delta Lake em destinos de dados

Usar formatos do Delta Lake em destinos de dados do Data Catalog

1. No menu Destino, escolha AWS Glue Studio Data Catalog.
2. Na guia Propriedades da fonte de dados, escolha um banco de dados e uma tabela.
3. O AWS Glue Studio exibe o tipo de formato como Data Lake e o URL do Amazon S3.

Usar formatos do Delta Lake em fontes de dados do Amazon S3

Insira os valores ou selecione-os entre as opções disponíveis para configurar o formato do Delta Lake.

- Tipo de compactação: escolha uma das opções de tipo de compactação: não compactado ou Snappy.
- Local de destino do Amazon S3: escolha o local de destino do Amazon S3 clicando em Procurar no S3.
- Opções de atualização do Data Catalog: atualizar o Data Catalog não é compatível com esse formato no editor visual do Glue Studio.
  - Do not update the Data Catalog (Não atualizar o Data Log): (padrão) escolha essa opção se você não quiser que o trabalho atualize o Data Catalog, mesmo que o esquema seja alterado ou novas partições sejam adicionadas.
  - Para atualizar o catálogo de dados após a execução do trabalho do AWS Glue, execute ou agende um crawler do AWS Glue. Para obter mais informações, consulte Como especificar opções de configuração para um armazenamento de dados do Delta Lake.
- Chaves de partição: escolha quais colunas serão usadas como chaves de particionamento na saída. Para adicionar mais chaves de partição, escolha Adicionar uma chave de partição.
- Opcionalmente, escolha Opções adicionais para inserir um par chave-valor. Por exemplo, um par de chave-valor pode ser: chave: timestampAsOf e valor: 2023-02-24 14:16:18.

Usar a estrutura do Apache Iceberg no AWS Glue Studio

Usar a estrutura do Apache Iceberg em destinos de dados

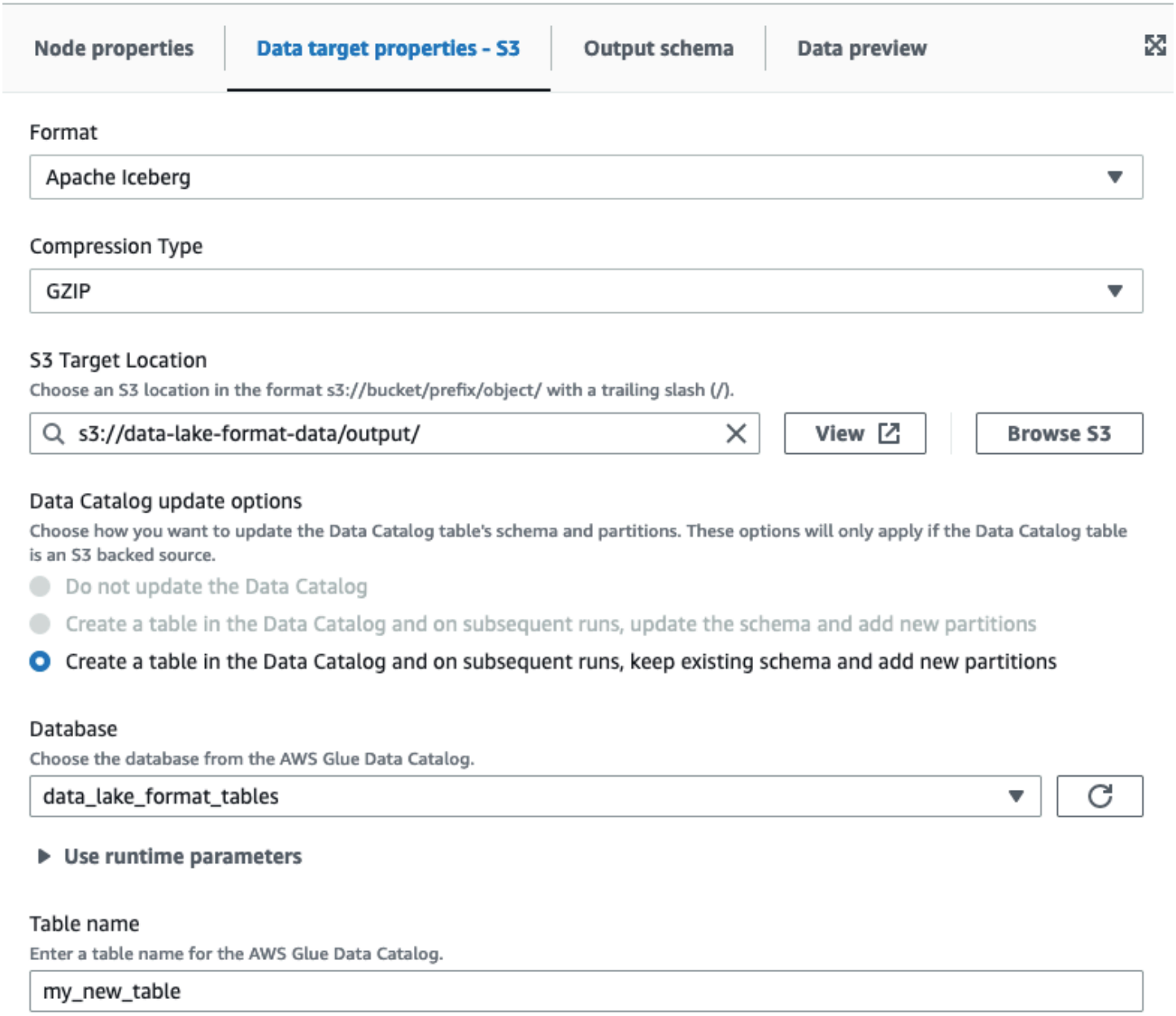
Usar a estrutura do Apache Iceberg em destinos de dados do Data Catalog

1. No menu Destino, escolha AWS Glue Studio Data Catalog.
2. Na guia Propriedades da fonte de dados, escolha um banco de dados e uma tabela.
3. O AWS Glue Studio exibe o tipo de formato como Apache Iceberg e o URL do Amazon S3.

Usar a estrutura do Apache Hudi em destinos de dados do Amazon S3

Insira os valores ou selecione-os entre as opções disponíveis para configurar o formato do Apache Iceberg.

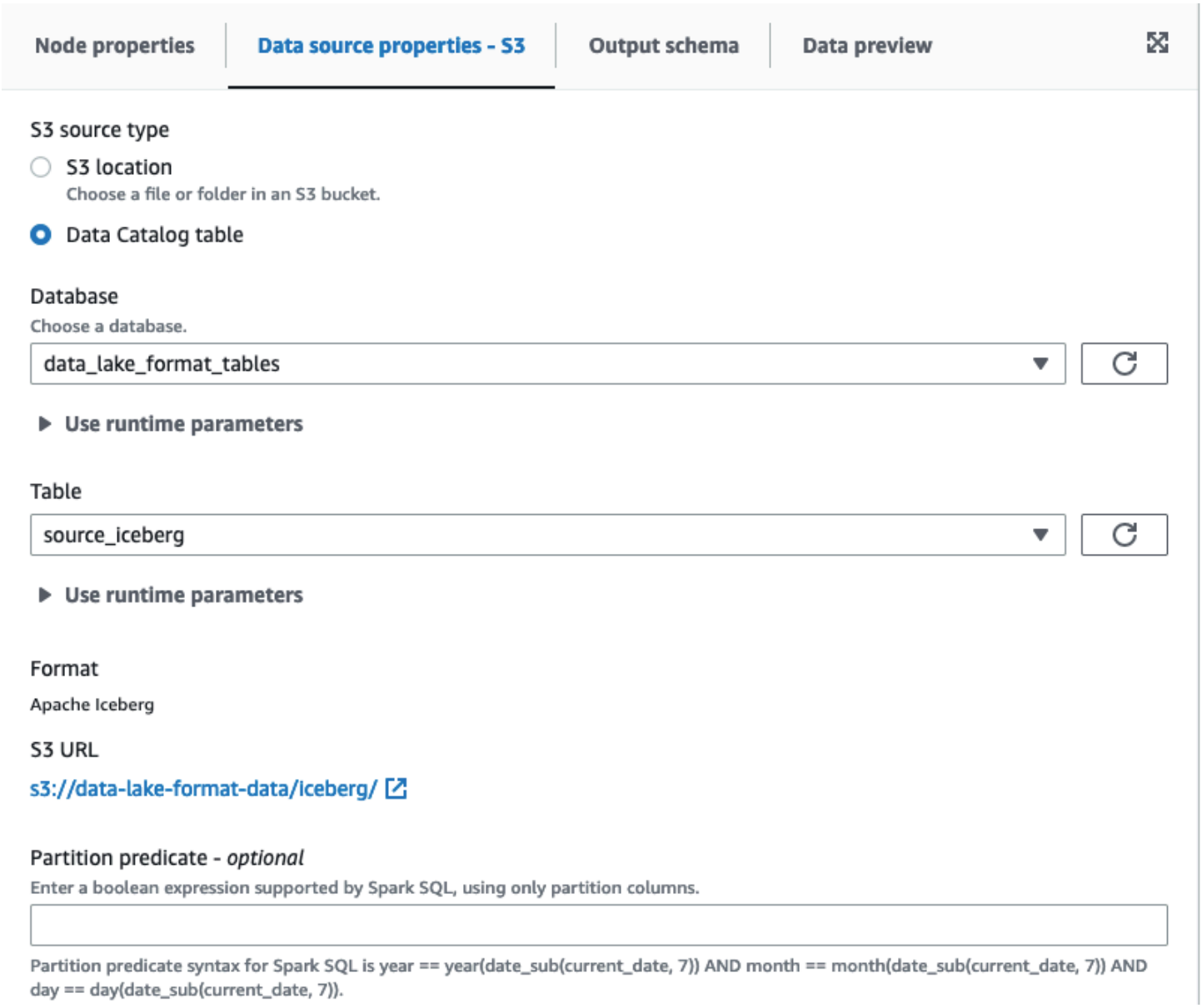
- Formatar: escolha Apache Iceberg no menu suspenso.
- Local de destino do Amazon S3: escolha o local de destino do Amazon S3 clicando em Procurar no S3.
- Opções de atualização do catálogo de dados:Criar uma tabela no catálogo de dados e em execuções subsequentes, manter o esquema existente e adicionar novas partições deve ser selecionado para continuar. Escrever uma nova tabela do Iceberg usando o AWS Glue requer que o Data Catalog seja configurado como o catálogo para a tabela do Iceberg. Para atualizar uma tabela do Iceberg existente que tenha sido registrada no Data Catalog, escolha Data Catalog como alvo.
  - Banco de dados: escolha o banco de dados do Data Catalog.
  - Nome da tabela: insira o valor para o nome da tabela. Os nomes das tabelas do Apache Iceberg devem estar em letras minúsculas. Use sublinhas se necessário, pois espaços não são permitidos. Por exemplo, "data_lake_format_tables".



Usar a estrutura do Apache Iceberg em fontes de dados do Amazon S3

Usar a estrutura do Apache Iceberg em fontes de dados do Data Catalog

1. No menu Fonte, escolha AWS Glue Studio Data Catalog.
2. Na guia Propriedades da fonte de dados, escolha um banco de dados e uma tabela.
3. O AWS Glue Studio exibe o tipo de formato como Apache Iceberg e o URL do Amazon S3.



Usar a estrutura do Apache Iceberg em fontes de dados do Amazon S3

O Apache Iceberg não está disponível como opção de dados para os nós de origem do Amazon S3 no AWS Glue Studio.

Configurar nós de destino de dados

O destino de dados é onde o trabalho grava os dados transformados.

Visão geral das opções de destino de dados

Seu destino de dados (também chamado de coletor de dados) pode ser:

- S3: o trabalho grava os dados em um arquivo no local do Amazon S3 escolhido e no formato especificado.

  Se você configurar colunas de partição para o destino de dados, o trabalho grava o conjunto de dados no Amazon S3 em diretórios com base na chave de partição.
- AWS Glue Data Catalog: o trabalho usa as informações associadas à tabela no Data Catalog para gravar os dados de saída em um local de destino.

  É possível criar a tabela manualmente ou com o crawler. Você também pode usar modelos do AWS CloudFormation para criar tabelas no Data Catalog.
- Um conector: um conector é um pedaço de código que facilita a comunicação entre o armazenamento de dados e o AWS Glue. O trabalho usa o conector e a conexão associada para gravar os dados de saída em um local de destino. Você pode assinar um conector oferecido no AWS Marketplace ou pode criar seu próprio conector personalizado. Para obter mais informações, consulte Adição de conectores ao AWS Glue Studio

Você pode optar por atualizar o Data Catalog quando seu trabalho grava em um destino de dados do Amazon S3. Em vez de exigir que um crawler atualize o Data Catalog quando o esquema ou as partições mudam, essa opção facilita a manutenção das tabelas atualizadas. Essa opção simplifica o processo de disponibilização de dados para análise, adicionando opcionalmente novas tabelas ao Data Catalog e atualizando partições de tabela e o esquema de tabelas diretamente a partir do trabalho.

Editar o nó de destino de dados

O destino de dados é onde o trabalho grava os dados transformados.

Para adicionar ou configurar um nó de destino de dados em seu diagrama de trabalho

1. (Opcional) se você precisar adicionar um nó de destino, escolha Target (Destino) na barra de ferramentas no topo do editor visual e escolha S3 ou Glue Data Catalog.

   - Se escolher S3 para o destino, o trabalho grava o conjunto de dados em um ou mais arquivos no local do Amazon S3 especificado.
   - Se escolher AWS Glue Data Catalog para o destino, o trabalho grava em um local descrito pela tabela selecionada no Data Catalog.
2. Escolha um nó de destino dos dados no diagrama de trabalho. Quando você escolhe um nó, o painel de detalhes do nó aparece no lado direito da página.

3. Escolha a guia Node properties (Propriedades do nó) e insira as seguintes informações:

   - Name (Nome): insira um nome a ser associado ao nó no diagrama de trabalho.
   - Node type (Tipo de nó): um valor já deve estar selecionado, mas você pode alterá-lo conforme necessário.
   - Node parents (Nós pais): o nó pai é o nó no diagrama de trabalho que fornece os dados de saída que você deseja gravar no local de destino. Para um diagrama de trabalho pré-preenchido, o nó de destino já deve ter o nó pai selecionado. Se não houver nenhum nó pai exibido, escolha um nó pai na lista.

     Um nó de destino tem um único nó pai.
4. Configure as informações de Data target properties (Propriedades do destino dos dados). Para obter mais informações, consulte as seções a seguir:

   - Uso do Amazon S3 para o destino de dados
   - Usar as tabelas do Data Catalog para o destino dos dados
   - Usar um conector para o destino de dados
5. (Opcional) depois de configurar as propriedades do nó de destino, você pode visualizar o esquema de saída para seus dados escolhendo a guia Output schema (Esquema de saída) no painel de detalhes do nó. Na primeira vez que você escolher essa guia para qualquer nó em seu trabalho, você receberá uma solicitação para fornecer uma função do IAM para acessar os dados. Se você não tiver especificado uma função do IAM na guia Job details (Detalhes do trabalho), você receberá uma solicitação para inserir uma função do IAM aqui.

Uso do Amazon S3 para o destino de dados

Para todas as origens dos dados, exceto o Amazon S3 e os conectores, uma tabela deve existir noAWS Glue Data Catalog para o tipo de origem que você escolher. O AWS Glue Studio não cria a tabela do Data Catalog.

Para configurar um nó de destino de dados que grava no Amazon S3

1. Vá para o editor visual para um trabalho novo ou salvo.
2. Escolha um nó de origem dos dados no diagrama de trabalho.
3. Escolha a guia Data source properties (Propriedades da origem dos dados) e insira as seguintes informações:

- Format (Formato): escolha um formato na lista. Os tipos de formato disponíveis para os resultados dos dados são:
  - JSON: JavaScript Object Notation.
  - CSV: Comma-separated values (valores separados por vírgula).
  - Avro: JSON binário do Apache Avro.
  - Parquet: Armazenamento em colunas no Apache Parquet.
  - Glue Parquet: um tipo de gravador Parquet personalizado, otimizado como o formato de dados para `DynamicFrames`. Em vez de exigir um esquema pré-calculado para os dados, ele calcula e modifica o esquema dinamicamente.
  - ORC: formato colunar de linha otimizado (ORC) do Apache.

  Para saber mais sobre essas opções de formato, consulte Opções de formato para entradas e saídas de ETL no AWS Glue no Guia do desenvolvedor do AWS Glue.
- Compression Type (Tipo de compactação): você pode optar por compactar os dados opcionalmente, usando o formato `gzip` ou `bzip2`. O padrão é sem compactação, ou None (Nenhum).
- S3 Target Location (Local de destino do S3): o bucket do Amazon S3 e o local para a saída de dados. Você pode selecionar o botão Browse S3 (Procurar no S3) para ver os buckets do Amazon S3 aos quais você tem acesso e escolher um como destino.
- Opções de atualização do Data Catalog
  - Do not update the Data Catalog (Não atualizar o Data Log): (padrão) escolha essa opção se você não quiser que o trabalho atualize o Data Catalog, mesmo que o esquema seja alterado ou novas partições sejam adicionadas.
  - Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions (Criar uma tabela no Data Catalog e em execuções subsequentes, atualizar o esquema e adicionar novas partições): se você escolher essa opção, o trabalho criará a tabela no Data Catalog na sua primeira execução. Em execuções de trabalho subsequentes, ele atualizará a tabela do Data Catalog, se o esquema for alterado ou novas partições forem adicionadas.

    Você também deve selecionar um banco de dados no Data Catalog e inserir um nome de tabela.
  - Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions (Criar uma tabela no Data Catalog e em execuções subsequentes, manter

o esquema atual e adicionar novas partições): se você escolher essa opção, o trabalho criará a tabela no Data Catalog na sua primeira execução. Em execuções de trabalho subsequentes, ele atualizará a tabela do Data Catalog, mas apenas para adicionar novas partições.

Você também deve selecionar um banco de dados no Data Catalog e inserir um nome de tabela.

- Partition keys (Chaves de partição): escolha quais colunas usar como chaves de particionamento na saída. Para adicionar mais chaves de partição, escolha Add a partition key (Adicionar uma chave de partição.

Usar as tabelas do Data Catalog para o destino dos dados

Para todas as origens dos dados, exceto o Amazon S3 e os conectores, uma tabela deve existir noAWS Glue Data Catalog para o tipo de destino que você escolher. O AWS Glue Studio não cria a tabela do Data Catalog.

Para configurar as propriedades de dados para um destino que usa uma tabela do Data Catalog

1. Vá para o editor visual para um trabalho novo ou salvo.
2. Escolha um nó de destino dos dados no diagrama de trabalho.
3. Selecione a guia Data target properties (Propriedades do destino de dados) e insira as seguintes informações:

   - Database (Banco de dados): escolha na lista o banco de dados que contém a tabela que você deseja usar como destino. Esse banco de dados já deve existir no Data Catalog.
   - Table (Tabela): escolha na lista a tabela que define o esquema de dados de saída. Essa tabela já deve existir no Data Catalog.

     Uma tabela no Data Catalog consiste em nomes de colunas, definições de tipos de dados, informações de partição e outros metadados relacionados a um conjunto de dados de destino. O trabalho grava em um local descrito por essa tabela no Data Catalog.

     Para obter mais informações sobre como criar tabelas no Data Catalog, consulte Definir tabelas no Data Catalog no Guia do desenvolvedor do AWS Glue.
   - Opções de atualização do Data Catalog

- Do not change table definition (Não alterar a definição da tabela): (padrão) escolha essa opção se não quiser que o trabalho atualize o Data Catalog, mesmo que o esquema seja alterado ou novas partições sejam adicionadas.
- Update schema and add new partitions (Atualizar esquema e adicionar novas partições): se você escolher essa opção, o trabalho atualizará a tabela do Data Catalog, se o esquema for alterado ou novas partições forem adicionadas.
- Keep schema and add new partitions (Manter o esquema e adicionar novas partições): se você escolher essa opção, o trabalho atualizará a tabela do Data Catalog, mas apenas para adicionar novas partições.
- Partition keys (Chaves de partição): escolha quais colunas usar como chaves de particionamento na saída. Para adicionar mais chaves de partição, escolha Add a partition key (Adicionar uma chave de partição.

Usar um conector para o destino de dados

Se você selecionar um conector como Node type (Tipo de nó), siga as instruções em Criação de trabalhos com conectores personalizados para concluir a configuração das propriedades do destino dos dados.

Editar ou carregar um script de trabalho

Use o editor visual do AWS Glue Studio para editar o script de trabalho ou carregar seu próprio script.

É possível usar o editor visual para editar os nós do trabalho somente quando os trabalhos foram criados com o AWS Glue Studio. Se o trabalho foi criado usando o console do AWS Glue, por meio de comandos de API, ou com a command line interface (CLI), você pode usar o editor de scripts no AWS Glue Studio para editar o script de trabalho, os parâmetros e a programação. Você também pode editar o script para um trabalho criado no AWS Glue Studio convertendo o trabalho para o modo somente script.

Para editar o script de trabalho ou carregar seu próprio script

1. Se estiver criando um novo trabalho, na página Jobs (Trabalhos), escolha a opção Spark script editor (Editor de scripts do Spark) para criar um trabalho do Spark ou escolha Pythin Shell script editor (Editor de scripts de shell do Python) para criar um trabalho de shell do Python. Você pode escrever um novo script ou carregar um existente. Se escolher Spark script editor

(Editor de scripts do Spark), você pode escrever ou carregar um script em Scala ou Python. Se escolher Python Shell script editor (Editor de scripts de shell do Python), você só pode escrever ou carregar um script em Python.

Depois de escolher a opção para criar um novo trabalho, na seção Options (Opções) que aparece, você pode optar por iniciar com um script inicial (Create a new script with boilerplate code [Criar um novo script com o código clichê]) ou pode carregar um arquivo local para usar como script de trabalho.

Se escolher Spark script editor (Editor de scripts do Spark), poderá carregar arquivos de script em Scala ou Python. Os scripts em Scala devem ter a extensão de arquivo `.scala`. Os scripts em Python devem ser reconhecidos como arquivos do tipo Python. Se escolher Python Shell script editor (Editor de scripts de shell do Python), só poderá carregar arquivos de script em Python.

Quando terminar de fazer suas escolhas, selecione Create (Criar) para criar o trabalho e abrir o editor visual.

2. Vá para o editor de trabalho visual para o trabalho novo ou salvo e escolha a guia Script.
3. Se você não tiver criado um novo trabalho usando uma das opções do editor de scripts e jamais tiver editado o script para um trabalho existente, a guia Script exibe o cabeçalho Script (Locked) (Script [bloqueado]). Isso significa que o editor de scripts está em modo somente leitura. Escolha Edit script (Editar script) para desbloquear o script para edição.

   Para tornar o script editável, o AWS Glue Studio converte seu trabalho de um trabalho visual para um trabalho somente de script. Se você desbloquear o script para edição, não poderá mais usar o editor visual para esse trabalho depois de salvá-lo.

   Na janela de confirmação, escolha Confirm (Confirmar) para continuar ou Cancel (Cancelar) para manter o trabalho disponível para edição visual.

   Se escolher Confirm (Confirme), a guia Visual não aparecerá mais no editor. Você pode usar o AWS Glue Studio para modificar o script usando o editor de scripts, modificar os detalhes ou programação do trabalho ou exibir execuções de trabalho.

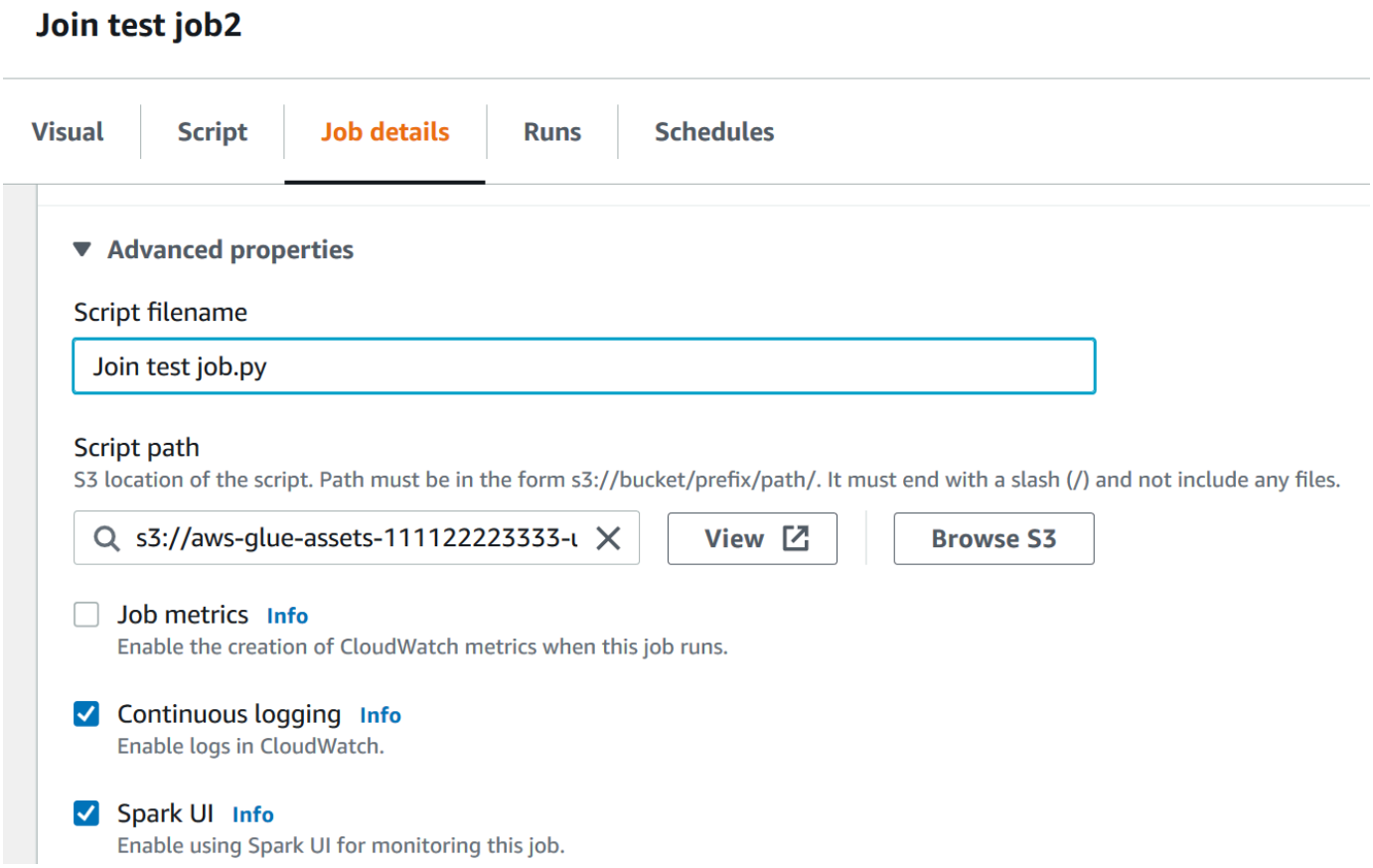
   > **Note**
   >
   > Até que você salve o trabalho, a conversão para um trabalho somente de script não é permanente. Se você atualizar a página da Web do console ou fechar o trabalho antes

de salvá-lo e reabri-lo no editor visual, ainda poderá editar os nós individuais no editor visual.

4. Edite o script, conforme necessário.

   Quando terminar de editar o script, escolha Save (Salvar) para salvar o trabalho e convertê-lo permanentemente de visual para somente script.

5. (Opcional) você pode baixar o script no console do AWS Glue Studio escolhendo o botão Download (Baixar) na guia Script. Quando você escolhe esse botão, uma nova janela do navegador é aberta, exibindo o script em sua localização no Amazon S3. Os parâmetros de trabalho Script filename (Nome de arquivo do script) e Script path (Caminho do script) na guia Job details (Detalhes do trabalho) determinam o nome e a localização do arquivo de script no Amazon S3.



   Quando você salva o trabalho, o AWS Glue salva o script de trabalho no local especificado por esses campos. Se você modificar o arquivo de script nesse local no Amazon S3, o AWS Glue Studio carregará o script modificado na próxima vez que você editar o trabalho.

Criar e editar scripts em Scala no AWS Glue Studio

Quando você escolhe o editor de scripts para criar um trabalho, por padrão, a linguagem de programação de trabalho é definida como `Python 3`. Se você optar por escrever um novo script em vez de carregar um, o AWS Glue Studio iniciará um novo script com texto boilerplate escrito em Python. Se em vez disso, você quiser escrever um script em Scala, deverá primeiro configurar o editor de scripts para usar o Scala.

> **Note**
>
> Se você escolher o Scala como a linguagem de programação para o trabalho e usar o editor visual para projetar seu trabalho, o script de trabalho gerado será escrito em Scala e nenhuma outra ação será necessária.

Para escrever um novo script em Scala no AWS Glue Studio

1. Crie um novo trabalho escolhendo a opção Spark script editor (Editor de scripts do Spark).
2. Em Options (Opções), escolha Create a new script with boilerplate code (Criar um novo script com o código clichê).
3. Escolha a guia Job details (Detalhes do trabalho) e defina Language (Linguagem) como `Scala` (em vez de `Python 3`).

   > **Note**
   >
   > A propriedade Type (Tipo) do trabalho é automaticamente definida como `Spark` quando você escolhe a opção Spark script editor (Editor de scripts do Spark) para criar um trabalho.

4. Escolha a guia Script.
5. Remova o texto clichê em Python. Você pode substituí-lo com o seguinte texto clichê em 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. Escreva seu script de trabalho em Scala no editor. Adicione mais declarações `import`, conforme necessário.

Criar e editr trabalhos de shell do Python no AWS Glue Studio

Quando você escolhe o editor de scripts de shell do Python para criar um trabalho, pode carregar um script em Python existente ou escrever um novo. Se você optar por escrever um novo script, o código clichê será adicionado ao novo script de trabalho em Python.

Para criar um novo trabalho de shell do Python

Consulte as instruções em Iniciar trabalhos no AWS Glue Studio.

As propriedades de trabalho que são suportadas para trabalhos de shell do Python não são as mesmas que as suportadas para trabalhos do Spark. A lista a seguir descreve as alterações nos parâmetros de trabalho disponíveis para trabalhos de shell do Python na guia Job details (Detalhes do trabalho).

- A propriedade Type (Tipo) para o trabalho é automaticamente definida como `Python Shell` e não pode ser alterada.
- Em vez deLanguage (Linguagem), há uma propriedade Python version (Versão do Python) para o trabalho. No momento, os trabalhos de shell do Python criados no AWS Glue Studio usam o Python 3.6.
- A propriedade Glue version (Versão do Glue) não está disponível, porque não se aplica a trabalhos de shell do Python.
- Em vez de Worker type (Tipo de operador) e Number of workers (Número de operadores), uma propriedade Data processing units (Unidades de processamento de dados) é mostrada. Essa propriedade de trabalho determina quantas unidades de processamento de dados (DPUs) são consumidas pelo shell do Python ao executar o trabalho.
- A propriedade Job bookmark (Marcador de trabalho) não está disponível, porque não é suportado para trabalhos de shell do Python.

- Em Advanced properties (Propriedades avançadas), as propriedades a seguir não estão disponíveis para trabalhos de shell do Python.
  - Métricas de trabalho
  - Registro em log contínuo
  - Spark UI (IU do Spark) e Spark UI logs path (Caminho de logs de IU do Spark)
  - Dependent jars path (Caminho de arquivos jar dependentes), no cabeçalho Libraries (Bibliotecas)

Alterar os nós pais de um nó no diagrama de trabalho

Você pode alterar os pais de um nó para mover nós dentro do diagrama de trabalho ou para alterar uma origem dos dados de um nó.

Para alterar o nó pai

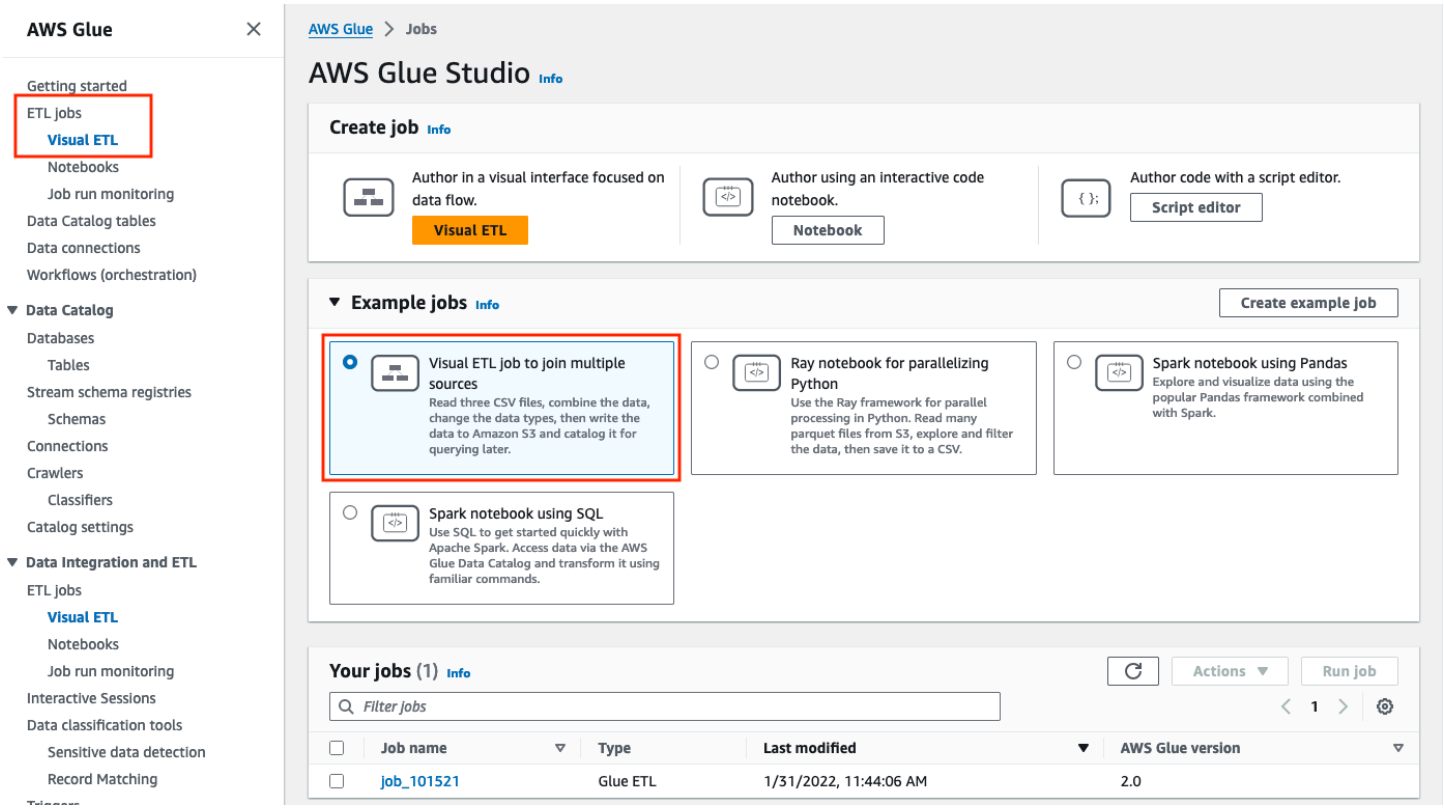
1. Escolha o nó no diagrama de trabalho que você deseja modificar.
2. No painel de detalhes do nó, na guia Node properties (Propriedades do nó) no cabeçalho Node parents (Nós pais), remova o pai atual do nó.
3. Escolha um novo nó pai na lista.
4. Modifique as outras propriedades do nó conforme necessário para corresponder ao nó pai recém-selecionado.

Se você modificou um nó por engano, pode usar o botão Undo (Desfazer) na barra de ferramentas para reverter a ação.

Excluir nós do diagrama de trabalho

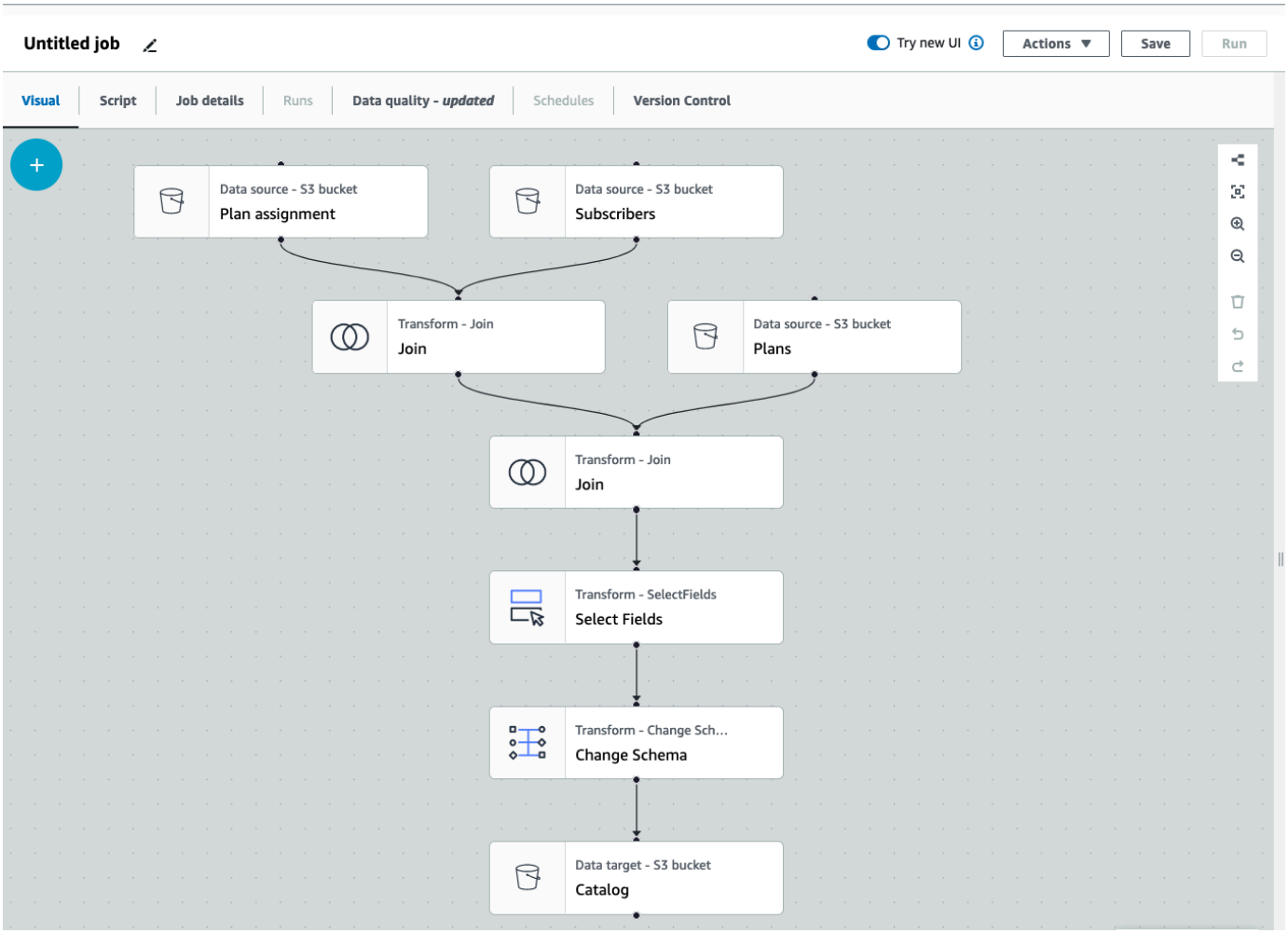
Ao trabalhar com trabalhos de ETL visuais, é possível remover nós da tela sem precisar adicionar novamente ou reestruturar nenhum nó conectado ao nó removido.

No exemplo abaixo, acompanhe escolhendo Trabalhos de ETL > ETL visual e, em Exemplos de trabalhos, escolhendo Trabalho de ETL visual para unir várias fontes. Escolha Criar trabalho de exemplo para criar um trabalho e siga as etapas abaixo.



Para remover um nó da tela

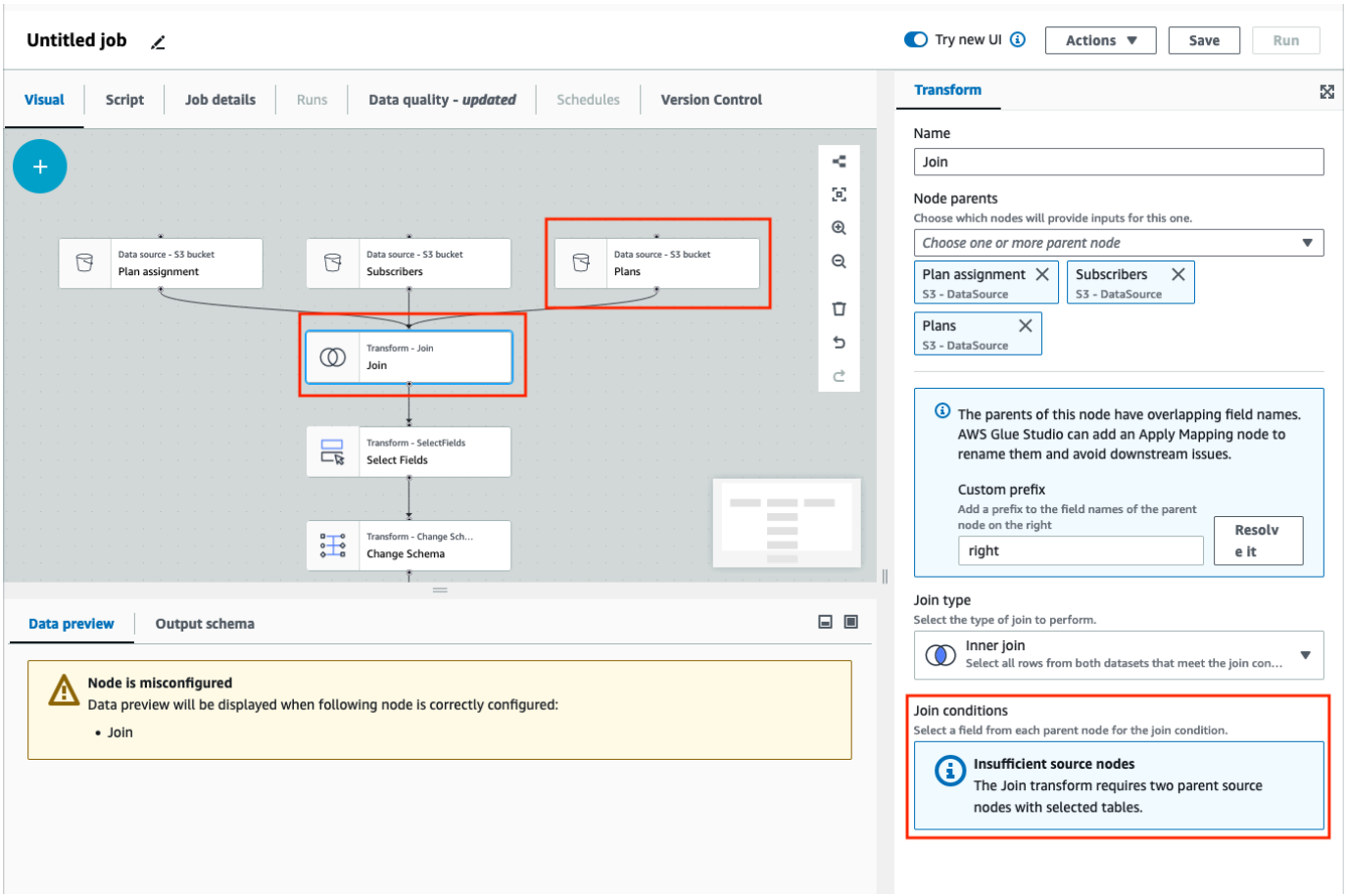
1. No console do AWS Glue, escolha ETL visual no menu de navegação e escolha uma tarefa existente. A tela de trabalho exibe o trabalho de exemplo conforme ilustrado abaixo.



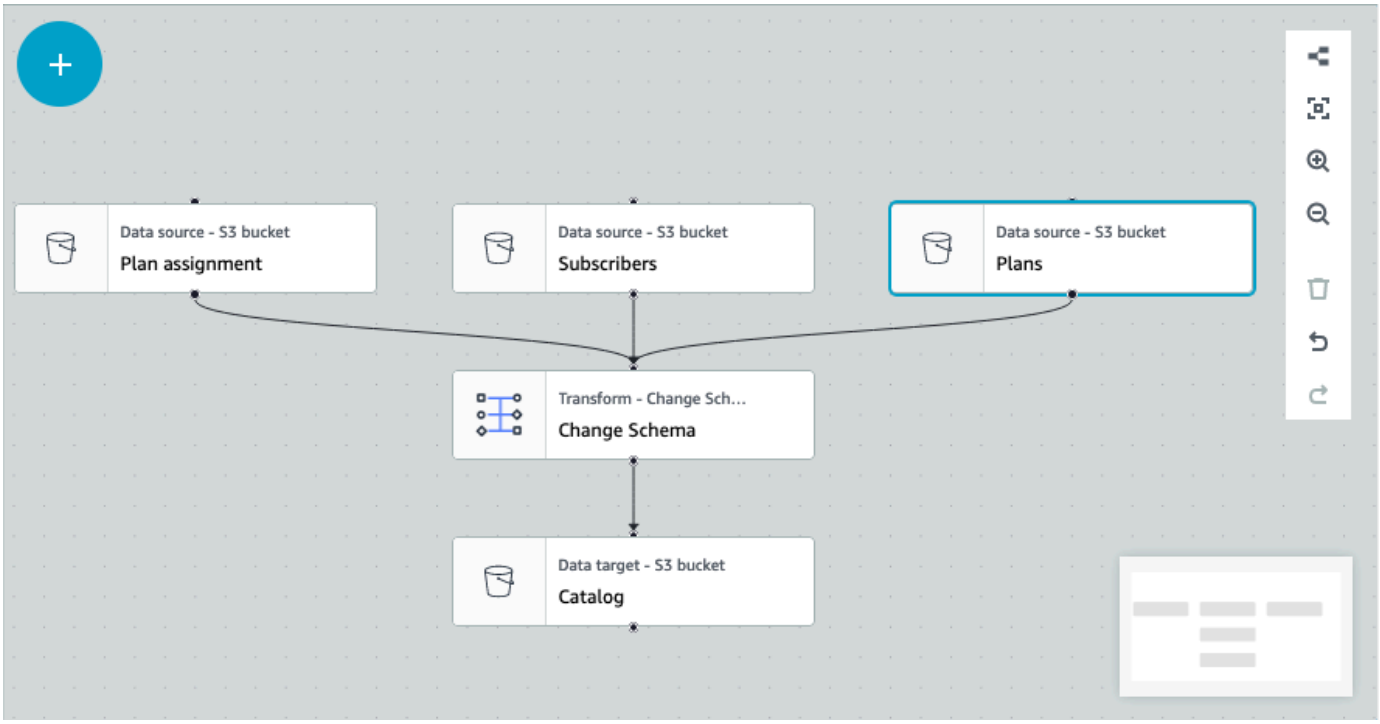
2. Escolha o nó que deseja remover. A tela ampliará o zoom no nó. Na barra de ferramentas no lado direito da tela, escolha o ícone Lixeira. Isso removerá o nó, e qualquer nó conectado ao nó será movido para ocupar seu lugar no fluxo de trabalho. Neste exemplo, o primeiro nó Join foi excluído da tela.

   Se você excluir um nó no fluxo de trabalho, o AWS Glue reorganizará os nós de uma forma que não resulte em um fluxo de trabalho inválido. Talvez você ainda precise corrigir a configuração de um nó.

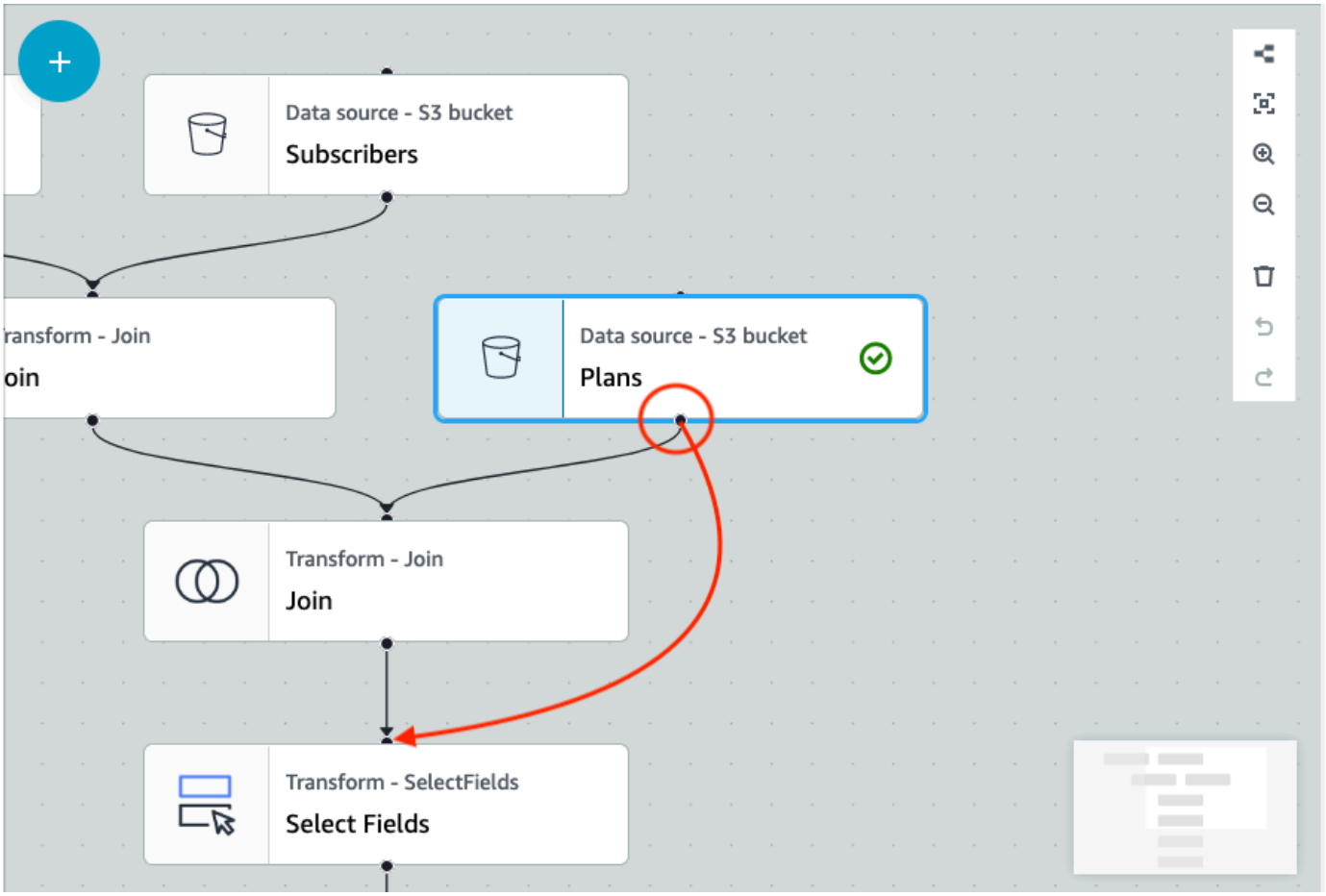
   No exemplo, o nó Join abaixo do nó Subscribers foi removido. Como resultado, o nó de origem Plans foi movido para o nível superior e ainda está conectado ao nó secundário Join. O nó Join agora requer configuração adicional, pois Join requer dois nós de origem pais com tabelas selecionadas. A guia Transformar à direita da tela exibe o requisito ausente em Condições de união.



3. Exclua o segundo nó Join e o nó Select Fields. Quando os nós forem excluídos, o fluxo de trabalho será semelhante ao exemplo abaixo.



4. Para modificar as conexões do nó, clique na alça do nó e arraste a conexão para um novo nó. Isso permitirá a você excluir os nós e reorganizá-los em um fluxo lógico. No exemplo, uma nova conexão está sendo feita com um clique na alça no nó Plans e arrastando-se a conexão até o nó Join, conforme ilustrado pela seta vermelha.



5. Se precisar desfazer alguma ação, escolha o ícone Desfazer diretamente abaixo do ícone Lixeira na barra de ferramentas no lado direito da tela.

Adicionar parâmetros de origem e destino ao nó do AWS Glue Data Catalog

O AWS Glue Studio permite que você parametrize trabalhos visuais. Como os nomes das tabelas de catálogo no ambiente de produção e desenvolvimento podem ser diferentes, você pode definir e selecionar parâmetros de runtime para bancos de dados e tabelas que serão executados quando seu trabalho for executado.

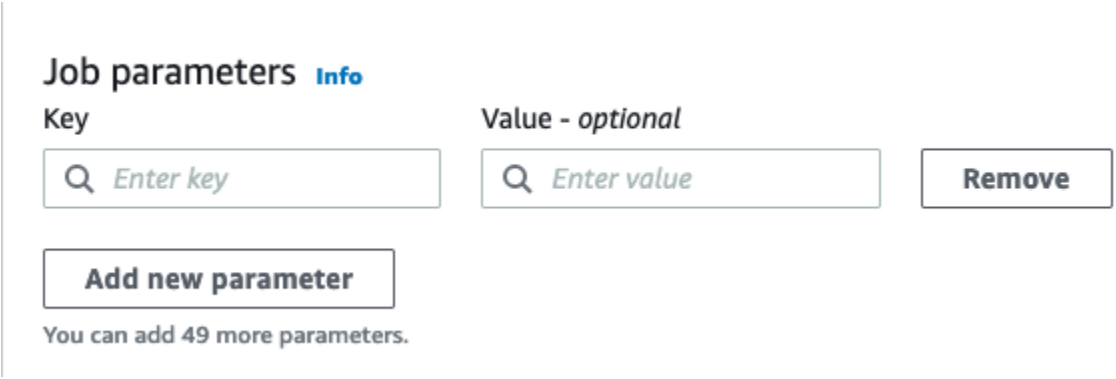
A parametrização de trabalhos permite parametrizar origens e destinos e salvar estes parâmetros no trabalho ao usar o nó do AWS Glue Data Catalog. Ao especificar origens e destinos como parâmetros, você está permitindo a reutilização de trabalhos, principalmente quando usa o mesmo trabalho em vários ambientes. Isso é útil para promover código em ambientes de implantação, economizando tempo e esforços no gerenciamento das suas origens e dos seus destinos. Além

disso, os parâmetros personalizados que você especificar substituirão os argumentos padrão para execuções específicas de trabalhos do AWS Glue.

Para adicionar parâmetros de origem e destino

Se você estiver usando o nó do AWS Glue Data Catalog como origem ou destino, poderá definir parâmetros de runtime na seção Advanced properties (Propriedades avançadas) na guia Job details (Detalhes do trabalho).

1. Escolha o nó do AWS Glue Data Catalog como nó de origem ou nó de destino.
2. Escolha a guia Job details (Detalhes do trabalho).
3. Escolha Advanced properties (Propriedades avançadas).
4. Na seção Job parameters (Parâmetros do trabalho), insira um valor de chave. Por exemplo, `--db.source` seria o parâmetro para uma origem de banco de dados. Você pode inserir qualquer nome para a chave, desde que o nome da chave seja seguido pelo “traço traço”.

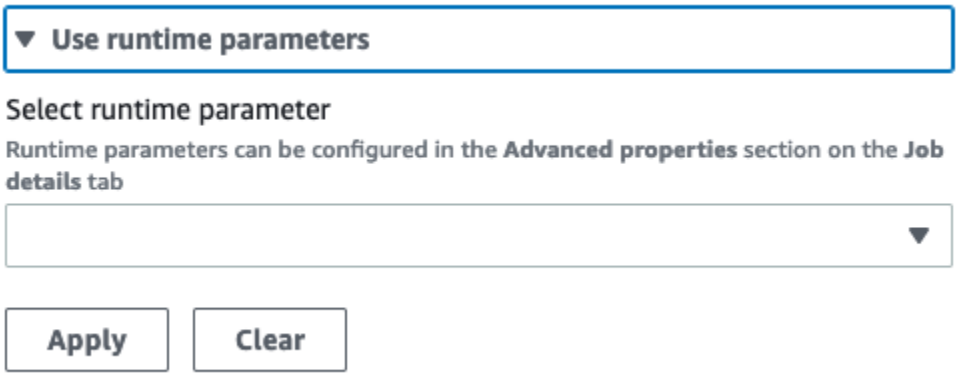


5. Insira o valor. Por exemplo, `databasename` seria o valor para um banco de dados sendo parametrizado.
6. Escolha Add new parameter (Adicionar novo parâmetro) se quiser adicionar parâmetros. O máximo de 50 parâmetros é permitido. Depois que o par de chave-valor tiver sido definido, você poderá usar o parâmetro no nó do AWS Glue Data Catalog.

Para selecionar um parâmetro de runtime

> **Note**
>
> O processo para selecionar parâmetros de runtime para bancos de dados e tabelas é o mesmo, não importando se o nó do AWS Glue Data Catalog é a origem ou o destino.

1. Escolha o nó do AWS Glue Data Catalog como nó de origem ou nó de destino.
2. Na guia Data source properties - Data Catalog (Propriedades da fonte de dados - Data Catalog) em Database (Banco de dados), escolha Use runtime parameters (Usar parâmetros de runtime).



3. No menu suspenso, escolha um parâmetro. Por exemplo, quando você selecionar um parâmetro definido para um banco de dados de origem, o banco de dados será preenchido automaticamente no menu suspenso de banco de dados quando você escolher Apply (Aplicar).
4. Na seção Table (Tabela), escolha um parâmetro que você já definiu como tabela de origem. Quando você escolhe Apply (Aplicar), a tabela é automaticamente preenchida como a tabela a ser usada.
5. Quando você salvar e executar o trabalho, o AWS Glue Studio fará referência aos parâmetros selecionados durante a execução do trabalho.

Usando sistemas de controle de versão do Git no AWS Glue

> **Note**
>
> Atualmente, os notebooks não são compatíveis com o controle de versão noAWS Glue Studio. No entanto, há suporte para controle de versão para scripts de AWS Glue tarefas e tarefas ETL visuais.

Se houver repositórios remotos e você quiser gerenciar seus trabalhos do AWS Glue usando seus repositórios, poderá usar o AWS Glue Studio ou a AWS CLI para sincronizar as alterações nos seus repositórios e seus trabalhos no AWS Glue. Quando sincroniza as alterações dessa forma, você está transferindo o trabalho do AWS Glue Studio para seu repositório ou transferindo do repositório para o AWS Glue Studio.

Com a integração do Git no AWS Glue Studio, você pode:

- Fazer a integração com sistemas de controle de versão do Git, como AWS CodeCommit, GitHub e Bitbucket
- Editar trabalhos do AWS Glue no AWS Glue Studio se você usa trabalhos visuais ou de script e os sincroniza com um repositório
- Parametrizar origens e destinos nos trabalhos
- Extrair trabalhos de um repositório e editá-los no AWS Glue Studio
- Testar trabalhos extraindo-os de ramificações e/ou transferindo-os para ramificações usando fluxos de trabalho em várias ramificações no AWS Glue Studio
- Baixar arquivos de um repositório e carregar trabalhos para o AWS Glue Studio para a criação de trabalhos entre contas
- Usar sua ferramenta de automação preferencial (por exemplo, Jenkins, AWS CodeDeploy etc.)

Este vídeo demonstra como você pode integrar o AWS ao Glue com Git e criar um pipeline de código contínuo e colaborativo.

Permissões do IAM

Certifique-se de que o trabalho tenha uma das seguintes permissões do IAM. Para obter mais informações sobre como configurar permissões do IAM, consulte Configurar permissões do IAM para o AWS Glue Studio.

- `AWSGlueServiceRole`
- `AWSGlueConsoleFullAccess`

No mínimo, as seguintes ações são necessárias para a integração com o Git:

- `glue:UpdateJobFromSourceControl`: para poder atualizar o AWS Glue com um trabalho presente em um sistema de controle de versão
- `glue:UpdateSourceControlFromJob`: para poder atualizar o sistema de controle de versão com um trabalho armazenado no AWS Glue
- `s3:GetObject`: para poder recuperar o script do trabalho durante a transferência para o sistema de controle de versão
- `s3:PutObject`: para poder atualizar o script ao extrair um trabalho de um sistema de controle de origem

Pré-requisitos

Para enviar trabalhos para um repositório de controle de origem, você precisará:

- um repositório que já foi criado pelo seu administrador
- uma ramificação no repositório
- um token de acesso pessoal (para o Bitbucket, esse é o token de acesso ao repositório)
- o nome de usuário do proprietário do repositório
- definir permissões no repositório para permitir AWS Glue Studio a leitura e gravação no repositório
  - GitLab — defina os escopos do token para api, read_repository e write_repository
  - Bitbucket — defina permissões para:
    - Associação ao Workspace — leia, escreva
    - Projetos — escrever, administrar ler
    - Repositórios — leitura, gravação, administração, exclusão

> **Note**
>
> Ao usar AWS CodeCommit, o token de acesso pessoal e o proprietário do repositório não são necessários. Consulte Conceitos básicos do Git e do AWS CodeCommit.

Usando trabalhos do seu repositório de controle de origem no AWS Glue Studio

Para extrair um trabalho do seu repositório de controle de origem que não está no AWS Glue Studio e usar esse trabalho no AWS Glue Studio, os pré-requisitos dependerão do tipo de trabalho.

Para um trabalho visual:

- você precisa de uma pasta e de um arquivo JSON da definição do trabalho que correspondam ao nome do trabalho

  Por exemplo, veja a definição de trabalho abaixo. A ramificação no seu repositório deve conter um caminho `my-visual-job/my-visual-job.json`, em que a pasta e o arquivo JSON correspondam ao nome do trabalho

```
{
  "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 um trabalho de script:

- você precisa de uma pasta, de um arquivo JSON da definição do trabalho e do script
- a pasta e o arquivo JSON devem corresponder ao nome do trabalho. O nome do script precisa corresponder ao `scriptLocation` da definição do trabalho junto com a extensão do arquivo

  Por exemplo, na definição do trabalho abaixo, a ramificação no seu repositório deve conter um caminho `my-script-job/my-script-job.json` e `my-script-job/my-script-job.py`. O nome do script deve corresponder ao nome na `scriptLocation`, incluindo a extensão do 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"
```

```
  }
}
```

Limitações

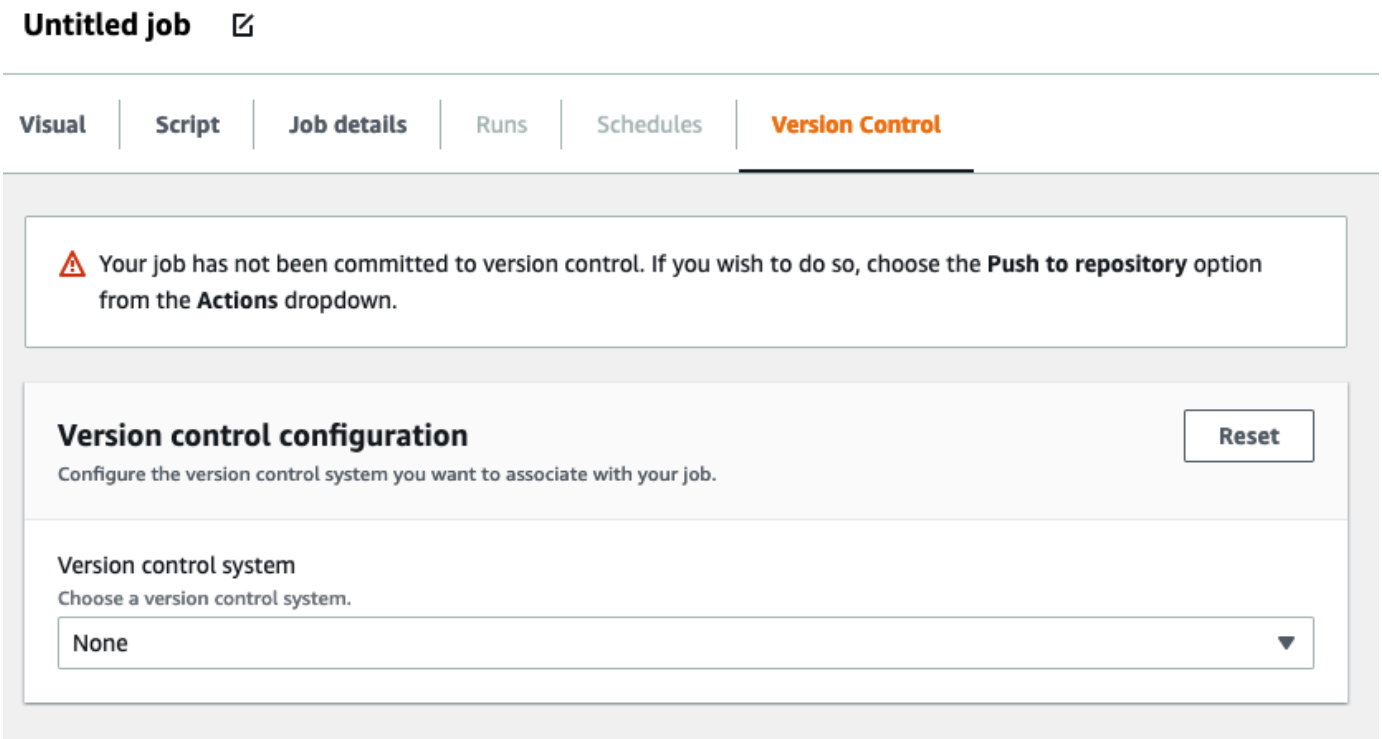
- No momento, o AWS Glue não oferece suporte a push/pull do GitLab-Groups.

Conectar repositórios de controle de versão com o AWS Glue

Você pode inserir os detalhes do seu repositório de controle de versão e gerenciá-los na guia Version Control (Controle de versão) no editor de trabalhos do AWS Glue Studio. Para se integrar ao seu repositório Git, deverá se conectar a ele para cada vez que fizer login no AWS Glue Studio.

Para conectar um sistema de controle de versão Git:

1. No AWS Glue Studio, comece um novo trabalho e escolha guia Version Control (Controle de versão).



2. Em Version control system (Sistema de controle de versão), escolha o serviço Git entre as opções disponíveis clicando no menu suspenso.

- AWS CodeCommit
- GitHub
- GitLab
- Bitbucket

3. Dependendo do sistema de controle de versão Git escolhido, você terá campos diferentes para preencher.

   para AWS CodeCommit:

   Conclua a configuração do repositório selecionando o repositório e a ramificação para seu trabalho:

   - Repositório: se você tiver configurado repositórios no AWS CodeCommit, selecione o repositório no menu suspenso. Seus repositórios serão preenchidos automaticamente na lista
   - Ramificação: selecione a ramificação no menu suspenso
   - Pasta: opcional - insira o nome da pasta na qual deseja salvar seu trabalho. Se o campo for deixado vazio, uma pasta será criada automaticamente. O nome da pasta assume o nome do trabalho como padrão.

   Para o GitHub:

   Conclua a configuração do GitHub preenchendo os campos:

   - Personal access token (Token de acesso pessoal): esse é o token fornecido pelo repositório do GitHub. Para obter mais informações sobre tokens de acesso pessoal, consulte GitHub Docs (Documentos do GitHub)
   - Repository owner (Proprietário do repositório): este é o proprietário do repositório do GitHub.

   Conclua a configuração do repositório selecionando o repositório e a ramificação no GitHub.

   - Repository (Repositório): se você tiver configurado repositórios no GitHub, selecione o repositório no menu suspenso. Seus repositórios serão preenchidos automaticamente na lista
   - Ramificação: selecione a ramificação no menu suspenso

- Pasta: opcional - insira o nome da pasta na qual deseja salvar seu trabalho. Se o campo for deixado vazio, uma pasta será criada automaticamente. O nome da pasta assume o nome do trabalho como padrão.

Para o GitLab:

> **Note**
>
> No momento, o AWS Glue não oferece suporte a push/pull do GitLab-Groups.

- Personal access token (Token de acesso pessoal): esse é o token fornecido pelo repositório do GitLab. Para obter mais informações sobre tokens de acesso pessoal, consulte Tokens de acesso pessoal do GitLab
- Repository owner (Proprietário do repositório): este é o proprietário do repositório do GitLab.

Conclua a configuração do repositório selecionando o repositório e a ramificação no GitLab.

- Repository (Repositório): se você tiver configurado repositórios no GitLab, selecione o repositório no menu suspenso. Seus repositórios serão preenchidos automaticamente na lista
- Ramificação: selecione a ramificação no menu suspenso
- Pasta: opcional - insira o nome da pasta na qual deseja salvar seu trabalho. Se o campo for deixado vazio, uma pasta será criada automaticamente. O nome da pasta assume o nome do trabalho como padrão.

Para Bitbucket:

- Senha de aplicações: o Bitbucket usa senhas de aplicações, e não tokens de acesso ao repositório. Para obter mais informações sobre senhas de aplicações, consulte Senhas de aplicações.
- Repository owner (Proprietário do repositório): este é o proprietário do repositório do Bitbucket. No Bitbucket, o proprietário é o criador do repositório.

Conclua a configuração do repositório selecionando o espaço de trabalho, repositório, ramificação e pasta do Bitbucket.

- Espaço de trabalho - se você tiver espaços de trabalho configurados no Bitbucket, selecione o espaço de trabalho no menu suspenso. Seus espaços de trabalho são preenchidos automaticamente
- Repository (Repositório) - se você tiver configurado repositórios no , selecione o repositório no menu suspenso. Seus repositórios são preenchidos automaticamente
- Ramificação: selecione a ramificação no menu suspenso. Suas filiais são preenchidas automaticamente
- Pasta: opcional - insira o nome da pasta na qual deseja salvar seu trabalho. Se deixado em branco, uma pasta será criada automaticamente com o nome do trabalho.

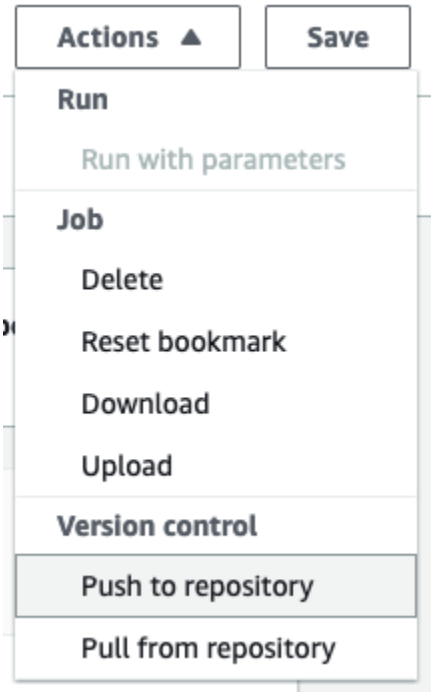
4. Escolha Save (Salvar) na parte superior do trabalho do AWS Glue Studio

Transferir trabalhos do AWS Glue para o repositório de origem

Depois de inserir os detalhes do seu sistema de controle de versão, você pode editar trabalhos no AWS Glue Studio e enviá-los para seu repositório de origem. Se você não estiver familiarizado com os conceitos do Git, como push e pull, consulte esse tutorial em Conceitos básicos do Git e do AWS CodeCommit.

Para transferir seu trabalho para um repositório, você precisa inserir os detalhes do seu sistema de controle de versão e salvar o trabalho.

1. No trabalho do AWS Glue Studio, escolha Actions (Ações). Opções adicionais de menu serão abertas.



2. Escolha Push to repository (Transferir para o repositório).

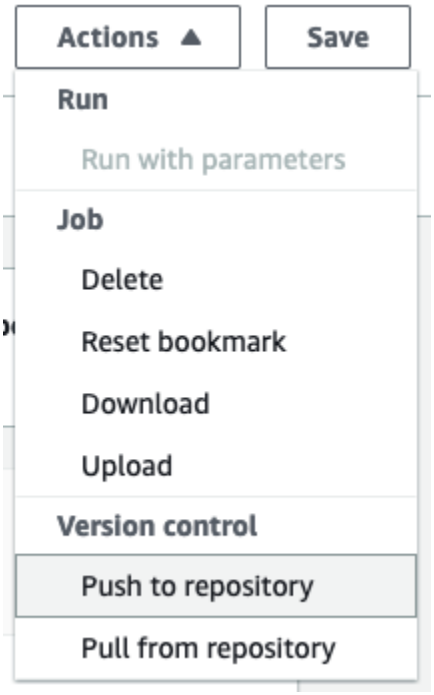
   Essa ação salvará o trabalho. Quando você transfere para o repositório, o AWS Glue Studio transfere a alteração mais recente salva. Se o trabalho no repositório tiver sido modificado por você ou por outro usuário e estiver fora de sincronia com o trabalho no AWS Glue Studio, o trabalho no repositório será substituído pelo trabalho salvo no AWS Glue Studio quando você transferir o trabalho do AWS Glue Studio.

3. Escolha Confirm (Confirmar) para concluir a ação. Uma nova confirmação será criada no repositório. Se você estiver usando o AWS CodeCommit, uma mensagem de confirmação exibirá um link para a confirmação mais recente no AWS CodeCommit.

Extrair trabalhos do AWS Glue do repositório de origem

Depois de inserir os detalhes do seu repositório do Git na guia Version control (Controle de versão), você também pode extrair trabalhos do seu repositório e editá-los no AWS Glue Studio.

1. No trabalho do AWS Glue Studio, escolha Actions (Ações). Opções adicionais de menu serão abertas.



2. Escolha Pull from repository (Extrair do repositório).
3. Selecione a opção Confirmar. Essa opção captura a confirmação mais recente do repositório e atualiza seu trabalho no AWS Glue Studio.
4. Edite seu trabalho no AWS Glue Studio. Se você fizer alterações, poderá sincronizar o trabalho com o repositório escolhendo Push to repository (Transferir para o repositório) no menu suspenso Actions (Ações).

Criação de código com notebooks AWS Glue Studio

Os engenheiros de dados podem criar trabalhos do AWS Glue de forma mais fácil e rápida do que antes usando a interface de caderno interativo no AWS Glue Studio ou sessões interativas no AWS Glue.

Tópicos

Visão geral do uso de cadernos

O AWS Glue Studio permite criar trabalhos interativamente em uma interface de caderno baseada no Jupyter Notebooks. Com os cadernos no AWS Glue Studio, você pode editar scripts de trabalho e visualizar a saída sem ter que executar um trabalho completo. Da mesma forma, é possível editar o código de integração de dados e visualizar a saída precisar executar um trabalho completo. Além disso, você pode adicionar markdowns e salvar cadernos como arquivos .ipynb e scripts de trabalho. É possível iniciar um caderno sem instalar software localmente nem gerenciar servidores. Quando estiver satisfeito com seu código, o AWS Glue Studio poderá converter seu caderno em um trabalho do Glue com o clique de um botão.

Alguns dos benefícios de usar cadernos incluem:

- Nenhum cluster para provisionar ou gerenciar
- Nenhum cluster ocioso para pagar
- Não é necessária nenhuma configuração inicial
- Não é necessário realizar a instalação de cadernos do Jupyter
- O mesmo runtime/plataforma que o ETL do AWS Glue.

Ao começar um caderno via AWS Glue Studio, todas as etapas de configuração são feitas para você, para que você possa explorar seus dados e começar a desenvolver seu script de trabalho após apenas alguns segundos. O AWS Glue Studio configura um caderno do Jupyter com o kernel Jupyter do AWS Glue. Não é necessário configurar VPCs, conexões de rede nem endpoints de desenvolvimento para usar esse caderno.

Para criar trabalhos usando a interface de caderno:

- configure as permissões do IAM necessárias.
- inicie uma sessão de caderno para criar um trabalho
- escreva código nas células do caderno
- execute e teste o código para visualizar a saída
- salve o trabalho

Depois que o caderno for salvo, ele será um trabalho do AWS Glue completo. Você pode gerenciar todos os aspectos do trabalho, como agendar trabalhos de execução, definir parâmetros de trabalho e exibir o histórico de execução do trabalho ao lado do seu caderno.

Criar um trabalho de ETL usando cadernos no AWS Glue Studio

Para começar a usar cadernos no console do AWS Glue Studio

1. Anexe políticas do AWS Identity and Access Management ao usuário AWS Glue Studio e crie um perfil do IAM para seu trabalho de ETL e bloco de anotações.
2. Configure a segurança adicional do IAM para cadernos, conforme descrito em Conceder permissões para a função do IAM.
3. Abra o console do AWS Glue Studio em https://console.aws.amazon.com/gluestudio/.

   > **Note**
   >
   > Verifique se o navegador não bloqueia cookies de terceiros. Qualquer navegador que bloqueie cookies de terceiros por padrão ou graças a uma configuração habilitada pelo usuário impedirá a inicialização dos cadernos. Para mais informações sobre o gerenciamento de cookies, consulte:

   - Chrome
   - Firefox
   - Safari
4. Escolha o link Jobs (Trabalhos) no menu de navegação do lado esquerdo.
5. Escolha Jupyter Notebook (Caderno do Jupyter) e, em seguida, escolha Create (Criar) para iniciar uma nova sessão de caderno.
6. Na página Create job in Jupyter notebook (Criar trabalho em caderno do Jupyter), forneça o nome do trabalho e escolha a função do IAM a ser usada. Escolha Criar trabalho.

   Após um curto período de tempo, o editor de caderno é mostrado.
7. Após adicionar o código, execute a célula para iniciar uma sessão. Há várias maneiras de executar a célula:

   - Pressione o botão play (reproduzir).
   - Use o atalho de teclado:
     - No macOS, Command+Enter para executar a célula.
     - No Windows, Shift+Enter para executar a célula.

Para obter informações sobre como escrever código usando uma interface do caderno do Jupyter, consulte a Documentação do usuário do Jupyter Notebook.

8. Para testar seu script, execute o script inteiro ou células individuais. Qualquer saída de comando será exibida na área abaixo da célula.
9. Após concluir o desenvolvimento de seu caderno, você poderá salvar o trabalho e executá-lo. Você encontrará o script na guia Script. Qualquer mágica adicionada ao caderno será removida e não será salva como parte do script do trabalho gerado do AWS Glue. O AWS Glue Studio adicionará automaticamente um `job.commit()` ao final do script gerado com base no conteúdo do caderno.

   Para obter mais informações sobre como executar trabalhos, consulte Iniciar uma execução de trabalho.

Componentes do editor de cadernos

A interface do editor de cadernos oferece as seguintes seções principais.

- Interface do caderno (painel principal) e barra de ferramentas
- Guias de edição de trabalhos

O editor de cadernos

O editor de cadernos do AWS Glue Studio é baseado na aplicação Jupyter Notebook. A interface de cadernos do AWS Glue Studio é semelhante à fornecida pelo Juypter Notebooks e é descrita na seção Interface de usuário de cadernos. O caderno usado por sessões interativas é um Jupyter Notebook.

Embora o caderno do AWS Glue Studio seja semelhante ao Juptyer Notebooks, ele apresenta algumas diferenças importantes:

- no momento, o caderno do AWS Glue Studio não pode instalar extensões
- não é possível usar várias guias; há uma relação 1:1 entre um trabalho e um caderno
- o caderno do AWS Glue Studio não tem o mesmo menu de arquivos superior que existe no Jupyter Notebooks

- no momento, o caderno do AWS Glue Studio só funciona com o kernel do AWS Glue. Observe que você não pode atualizar o kernel por conta própria.

Guias de edição de trabalhos do AWS Glue Studio

As guias que você usa para interagir com o trabalho ETL estão na parte superior da página do caderno. Elas são semelhantes às guias que aparecem no editor de trabalhos visual do AWS Glue Studio e executam as mesmas ações.

- Notebook (caderno): use essa guia para visualizar o script do trabalho usando a interface do caderno.
- Job details (Detalhes do trabalho): configure o ambiente e as propriedades para as execuções de trabalhos.
- Runs (Execuções): visualize informações sobre execuções anteriores deste trabalho.
- Schedules (Programações): configure uma programação para executar seu trabalho em horários específicos.

Salvar o caderno e o script do trabalho

Você pode salvar seu caderno e o script do trabalho que você está criando a qualquer momento. Basta escolher o botão Save (Salvar) no canto superior direito, o mesmo que você faria se estivesse usando o editor visual ou de script.

Quando você escolhe Save (Salvar), o arquivo do caderno é salvo nos locais padrão.

- Por padrão, o script do trabalho é salvo no local do Amazon S3 indicado na guia Job Details (Detalhes do trabalho), em Advanced properties (Propriedades avançadas), na propriedade dos detalhes do trabalho Script path (Caminho do script). Os scripts de trabalho são salvos em uma subpasta denominada `Scripts`.
- Por padrão, o arquivo do caderno (`.ipynb`) é salvo no local do Amazon S3 indicado na guia Job Details (Detalhes do trabalho), em Advanced properties (Propriedades avançadas), no Script path (Caminho do script) dos detalhes do trabalho. Os arquivos de cadernos são salvos em uma subpasta denominada `Notebooks`.

> **Note**
>
> Quando você salva o trabalho, o script de trabalho contém apenas as células de código do caderno. As células e magics do Markdown não estão incluídas no script de trabalho. Porém, o arquivo `.ipynb` conterá todas as remarcações e magics.

Após salvar o trabalho, você poderá executar o trabalho usando o script criado no caderno.

Gerenciar sessões de cadernos

Os cadernos no AWS Glue Studio são baseados no recurso de sessões interativas do AWS Glue. Há um custo para usar sessões interativas. Para ajudar a gerenciar seus custos, você pode monitorar as sessões criadas para sua conta e definir as configurações padrão para todas as sessões.

Alterar o tempo limite padrão para todas as sessões do caderno

Por padrão, o caderno provisionado do AWS Glue Studio expira após 12 horas se o caderno tiver sido iniciado e nenhuma célula executada. Não há custo associado a ele e o tempo-limite não é configurável.

Após executar uma célula, isso iniciará uma sessão interativa. Essa sessão tem um tempo limite padrão de 48 horas. Esse tempo limite pode ser configurado mediante a transmissão de uma mágica de `%idle_timeout` antes de executar uma célula.

Para modificar o tempo limite de sessão padrão para cadernos no AWS Glue Studio

1. No caderno, insira o magic `%idle_timeout` em uma célula e especifique o valor do tempo limite em minutos.
2. Por exemplo: `%idle_timeout 15` mudará o tempo limite padrão para 15 minutos. Se a sessão não for usada em 15 minutos, ela será interrompida automaticamente.

Instalar módulos Python adicionais

Se desejar instalar módulos adicionais em sua sessão usando pip, você poderá fazê-lo usando `%additional_python_modules`para adicioná-los à sua sessão:

```
%additional_python_modules awswrangler, s3://mybucket/mymodule.whl
```

Todos os argumentos para additional_python_modules são passados para `pip3 install -m <>`

Para visualizar uma lista de módulos Python disponíveis, consulte Usar bibliotecas Python com o AWS Glue.

Alterar a configuração do AWS Glue

É possível usar mágicas para controlar valores de configuração de trabalho do AWS Glue. Se quiser alterar um valor de configuração de trabalho, use a mágica adequada no caderno. Consulte Mágicas compatíveis com sessões interativas do AWS Glue para Jupyter.

> **Note**
>
> As propriedades de substituição de uma sessão em execução não estão mais disponíveis. Para alterar as configurações da sessão, você pode interromper a sessão, definir as novas configurações e, em seguida, iniciar uma nova sessão.

O AWS Glue oferece suporte a vários tipos de operadores. Você pode definir o tipo do operador com `%worker_type`. Por exemplo: `%worker_type G.2X` . O padrão é G.1X.

Também é possível especificar o número de operadores com `%number_of_workers`. Por exemplo, para especificar 40 operadores: `%number_of_workers 40`.

Para obter mais informações, consulte Definir as propriedades do trabalho

Interromper uma sessão de caderno

Para interromper uma sessão de caderno, use o magic `%stop_session`.

Se você navegar para longe do caderno no console do AWS, receberá uma mensagem de aviso onde poderá optar por interromper a sessão.

Usando CodeWhisperer com AWS Glue Studio notebooks

O AWS Glue Studio permite criar trabalhos interativamente em uma interface de caderno baseada no Jupyter Notebooks. O uso CodeWhisperer melhora a experiência de criação em AWS Glue Studio notebooks.

A CodeWhisperer extensão da Amazon oferece suporte à escrita de código gerando recomendações de código e sugerindo melhorias relacionadas a problemas de código.

O que é a Amazon CodeWhisperer?

A Amazon CodeWhisperer é um serviço baseado em aprendizado de máquina que ajuda a melhorar a produtividade do desenvolvedor. CodeWhisperer consegue isso gerando recomendações de código com base nos comentários dos desenvolvedores em linguagem natural e em seu código no IDE. Durante a versão prévia, a Amazon CodeWhisperer está disponível para Java, Python JavaScript, C# e TypeScript linguagens de programação. O serviço se integra com o Amazon SageMaker Studio JupyterLab, instâncias de Amazon SageMaker notebook e outros ambientes de desenvolvimento integrados (IDEs).

Para obter mais informações, consulte Configurando CodeWhisperer com AWS Glue Studio.

Status de execução de trabalhos do AWS Glue no console

Você pode visualizar o status de um trabalho de extração, transformação e carregamento (ETL) do AWS Glue enquanto ele está em execução ou após a interrupção. É possível visualizar o status usando o console do AWS Glue. Para obter mais informações sobre o status de execução de trabalhos, consulte the section called “Status de execução de trabalho”.

Acessar o painel de monitoramento de trabalhos

Acesse o painel de monitoramento de trabalhos escolhendo o link Monitoring (Monitoramento) no painel de navegação do AWS Glue.

Visão geral do painel de monitoramento de trabalhos

O painel de monitoramento de trabalhos fornece um resumo geral das execuções de trabalho, com totais para os trabalhos com um status Running (Executando), Canceled (Cancelado), Success (Bem-sucedido) ou Failed (Com falha). Os blocos adicionais fornecem a taxa geral de sucesso da execução do trabalho, o uso estimado de DPU para trabalhos, uma divisão das contagens de status de trabalhos por tipo de trabalho, tipo de operador e dia.

Os gráficos nos blocos são interativos. Você pode escolher qualquer bloco em um gráfico para executar um filtro que exiba apenas esses trabalhos na tabela Job runs (Execuções do trabalho) na parte inferior da página.

Você pode alterar o intervalo de datas para as informações exibidas nessa página usando o seletor Data range (Intervalo de datas). Quando você altera o intervalo de datas, os blocos de informações são ajustados a fim de exibir os valores para o número especificado de dias antes da data atual. Você também pode usar um intervalo de datas específico se escolher Custom (Personalizado) no seletor de intervalo de datas.

Visualizar execuções do trabalho

> **Note**
>
> O histórico de execução de trabalhos pode ser acessado por 90 dias para seu fluxo de trabalho e execução de trabalhos.

O recurso Job runs (Execuções do trabalho) exibe os trabalhos para o intervalo de datas e filtros especificados.

Você pode filtrar os trabalhos em critérios adicionais, como status, tipo de operador, tipo de trabalho e nome do trabalho. Na caixa de filtro na parte superior da tabela, você pode inserir o texto a ser usado como filtro. Os resultados da tabela são atualizados com linhas que contêm correspondências à medida que você digita o texto.

Você pode exibir um subconjunto dos trabalhos escolhendo elementos nos gráficos no painel de monitoramento de trabalho. Por exemplo, se você escolher o número de trabalhos em execução no bloco Job runs summary (Resumo de execuções do trabalho), a lista Job runs (Execuções do trabalho) exibe apenas os trabalhos que têm atualmente o status `Running`. Se você escolher uma das barras no gráfico de barras Worker type breakdown (Detalhamento do tipo de operador), somente as execuções de trabalho com o tipo de operador e status correspondentes serão mostrados na lista Job runs (Execuções do trabalho).

O recurso Job runs (Execuções do trabalho) exibe os detalhes das execuções do trabalho. É possível classificar as linhas na tabela escolhendo um cabeçalho de coluna. A tabela contém as seguintes informações:

Propriedade	Descrição
Nome do trabalho	O nome do trabalho do .
Tipo	O tipo de ambiente do trabalho:

Propriedade	Descrição
	• Glue ETL (ETL do Glue): execuções em um ambiente Apache Spark gerenciado pelo AWS Glue. • Glue Streaming (Transmissão do Glue): executa em um ambiente Apache Spark e realiza ETL em fluxos de dados. • Python shell: executa scripts do Python como um shell.
Horário de início	A data e a hora em que a execução deste trabalho foi iniciada.
End time (Horário de término)	A data e a hora em que a execução desse trabalho foi concluída.
Run status (Status da execução)	O estado atual da execução do trabalho. Os valores podem ser: • `STARTING` • `RUNNING` • `STOPPING` • `STOPPED` • `SUCCEEDED` • `FAILED` • `TIMEOUT`
Run time (runtime)	A quantidade de tempo em que a execução de trabalho consumiu recursos.

Propriedade	Descrição
Capacity	O número de unidades de processamento de dados (DPUs) do AWS Glue alocadas para essa execução de trabalho. Para obter mais informações sobre planejamento de capacidade, consulte Monitoramento de planejamento de capacidade de DPU no Guia do desenvolvedor do AWS Glue.

Propriedade	Descrição
Tipo de operador	O tipo de operador predefinido que é alocado quando um trabalho é executado. Os valores podem ser `G.1X`, `G.2X`, `G.4X` ou `G.8X`. • **`G.1X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador mapeia para 1 DPU (4 vCPUs, 16 GB de memória) com 84 GB de disco (aproxima damente 34 GB livres). Recomendamos esse tipo de operador para trabalhos com uso intensivo de memória. Esse é o Worker type (Tipo de operador) padrão para trabalhos do AWS Glue versão 2.0 ou posterior. • **`G.2X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador mapeia para 2 DPU (8 vCPUs, 32 GB de memória) com 128 GB de disco (aproxima damente 77 GB livres). Recomendamos esse tipo de operador para trabalhos com uso intensivo de memória e trabalhos que executem transformações de machine learning. • **`G.4X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador mapeia para 4 DPU (16 vCPUs, 64 GB de memória) com 256 GB de disco (aproxima damente 235 GB livres). Recomenda mos esse tipo de operador para trabalhos cujas workloads contêm as transform ações, agregações, uniões e consultas mais exigentes. Esse tipo de operador

Propriedade	Descrição
	está disponível somente para trabalhos de ETL do Spark no AWS Glue versão 3.0 ou posterior nas seguintes regiões da AWS: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central) , Europa (Frankfurt), Europa (Irlanda) e Europa (Estocolmo). • **`G.8X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador mapeia para 8 DPU (32 vCPUs, 128 GB de memória) com 512 GB de disco (aproxima damente 487 GB livres). Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de operador está disponível apenas para trabalhos ETL do Spark do AWS Glue versão 3.0 ou posterior , nas mesmas regiões da AWS compatíveis com o tipo de operador `G.4X`.
DPU hours (Horas de DPU)	O número estimado de DPUs usadas para a execução de trabalho. Uma DPU é uma medida relativa do poder de processamento. As DPUs são usadas para determinar o custo da execução de trabalho. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

É possível escolher qualquer execução de trabalho na lista e visualizar informações adicionais. Escolha uma execução de trabalho e realize uma das seguintes ações:

- Selecione o menu Actions (Ações) e a opção View job (Visualizar trabalho) para visualizar o trabalho no editor visual.
- Selecione o menu Actions (Ações) e a opção Stop run (Interromper execução) para interromper a execução atual do trabalho.
- Escolha o botão View CloudWatch logs (Visualizar logs do CloudWatch) para exibir os logs de execução do trabalho.
- Escolha Visualizar detalhes para visualizar a página de detalhes da execução do trabalho.

Visualizar os logs de execuções de trabalho

Você pode visualizar os logs de trabalhos de várias maneiras:

- Na página Monitoring (Monitoramento), na tabela Job runs (Execuções do trabalho), escolha uma execução de trabalho e selecione View CloudWatch logs (Visualizar logs do CloudWatch).
- No editor visual de trabalhos, na guia Runs (Execuções) de um trabalho, escolha os hiperlinks para exibir os logs:
  - Logs: links para os logs de trabalho do Apache Spark gravados quando o registro em log contínuo é habilitado para uma execução de trabalho. Quando você escolhe esse link, ele leva você para os logs do Amazon CloudWatch no grupo de logs `/aws-glue/jobs/logs-v2`. Por padrão, os logs excluem a pulsação do Apache Hadoop YARN e as mensagens de log do driver ou do executor do Apache Spark desnecessárias. Para obter mais informações sobre registro em log contínuo, consulte Registro em log contínuo para trabalhos do AWS Glue no Guia do desenvolvedor do AWS Glue.
  - Error logs (Logs de erro): vinculam-se aos logs gravados em `stderr` para a execução de trabalho. Quando você escolhe esse link, ele leva você para os logs do Amazon CloudWatch no grupo de logs `/aws-glue/jobs/error`. Você pode usar esses logs para exibir detalhes sobre os erros que foram encontrados durante a execução de trabalho.
  - Output logs (Logs de saída): links para os logs gravados em `stdout` para a execução de trabalho. Quando você escolhe esse link, ele leva você para os logs do Amazon CloudWatch no grupo de logs `/aws-glue/jobs/output`. Você pode usar esses logs para ver todos os detalhes sobre as tabelas que foram criadas no AWS Glue Data Catalog e os erros que foram encontrados.

Visualizar os detalhes de uma execução de trabalho

Você pode escolher um trabalho na lista Job runs (Execuções do trabalho) na página Monitoring (Monitoramento) e, em seguida, escolher View run details (Visualizar detalhes da execução) para ver informações detalhadas sobre a execução de trabalho.

As informações exibidas na página de detalhes da execução de trabalho incluem:

Propriedade	Descrição
Nome do trabalho	O nome do trabalho do .
Run status (Status da execução)	O estado atual da execução do trabalho. Os valores podem ser: • `STARTING` • `RUNNING` • `STOPPING` • `STOPPED` • `SUCCEEDED` • `FAILED` • `TIMEOUT`
Versão do Glue	A versão do AWS Glue usada pela execução de trabalho.
Recent attempt (Tentativa recente)	O número de tentativas automáticas de repetição da execução deste trabalho.
Horário de início	A data e a hora em que a execução deste trabalho foi iniciada.
End time (Horário de término)	A data e a hora em que a execução desse trabalho foi concluída.
Start-up time (Horário de início)	A quantidade de tempo gasta preparando para executar o trabalho.

Propriedade	Descrição
Tempo de execução	A quantidade de tempo gasta executando o script do trabalho.
Nome do gatilho	O nome do acionador associado ao trabalho.
Modificação mais recente	A data em que o trabalho foi modificado pela última vez.
Configuração de segurança	A configuração de segurança do trabalho, que inclui criptografia do Amazon S3, criptografia do CloudWatch e configurações de criptografia de marcadores de trabalho.
Timeout (Tempo limite)	O valor do tempo limite da execução de trabalho.
Allocated capacity (Capacidade alocada)	O número de unidades de processamento de dados (DPUs) do AWS Glue alocadas para essa execução de trabalho. Para obter mais informações sobre planejamento de capacidade, consulte Monitoramento de planejamento de capacidade de DPU no Guia do desenvolvedor do AWS Glue.
Max capacity (Capacidade máxima)	A capacidade máxima disponível para a execução de trabalho.
Número de operadores	O número de operadores usados para a execução de trabalho.

Propriedade	Descrição
Tipo de operador	O tipo de operadores predefinidos alocados para a execução de trabalho. Os valores podem ser `G.1X` ou `G.2X`. • **`G.1X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador é mapeado para 1 DPU (4 vCPUs, 16 GB de memória, disco de 64 GB), e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos com uso intensivo de memória. Esse é o Worker type (Tipo de operador) padrão para trabalhos do AWS Glue versão 2.0 ou posterior. • **`G.2X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador é mapeado para 2 DPUs (8 vCPUs, 32 GB de memória, disco de 128 GB), e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos com uso intensivo de memória e trabalhos que executem transformações de machine learning.
Logs	Um link para os logs de trabalho para registro em log contínuo (`/aws-glue/jobs/logs-v2`).
Output Logs (Logs de saída)	Um link para os arquivos de log de saída do trabalho (`/aws-glue/jobs/output`).
Logs de erro	Um link para os arquivos de log de erros do trabalho (`/aws-glue/jobs/error`).

Você também pode visualizar os seguintes itens adicionais, os quais se tornam disponíveis ao visualizar informações sobre execuções de trabalhos recentes. Para ter mais informações, consulte the section called “Exibir informações para execuções de trabalho recentes”.

- Argumentos de entrada
- Logs contínuos
- Métricas: é possível ver visualizações de métricas básicas. Para obter ter mais informações sobre as métricas incluídas, consulte the section called “Visualizar métricas do Amazon CloudWatch para uma execução de trabalho do Spark”.
- Interface do usuário do Spark: é possível visualizar os logs do Spark do seu trabalho na interface do usuário do Spark. Para obter mais informações sobre o uso da interface do usuário Web do spark , consulte o the section called “Monitorar com a interface do usuário do Spark”. Habilite esse recurso seguindo o procedimento em the section called “Habilitar a interface do usuário do Spark para trabalhos”.

Visualizar métricas do Amazon CloudWatch para uma execução de trabalho do Spark

Na página de detalhes de uma execução de trabalho, abaixo da seção Run details (Detalhes da execução), você pode visualizar as métricas do trabalho. O AWS Glue Studio envia métricas de trabalho para o Amazon CloudWatch para cada trabalho executado.

O AWS Glue relata as métricas ao Amazon CloudWatch a cada 30 segundos. As métricas do AWS Glue representam valores do delta a partir dos valores relatados anteriormente. Quando apropriado, os painéis de métricas agregam (somam) os valores de 30 segundos para obter um valor para o último minuto inteiro. No entanto, as métricas do Apache Spark que o AWS Glue transfere para o Amazon CloudWatch são geralmente valores absolutos que representam o estado atual no momento em que são relatadas.

> **Note**
>
> Você deve configurar sua conta para acessar o Amazon CloudWatch.

As métricas fornecem informações sobre a execução de trabalho, como:

- ETL Data Movement (Movimentação de dados de ETL): o número de bytes lidos ou gravados no Amazon S3.
- Memory Profile: Heap used (Perfil de memória: heap usado): o número de bytes de memória usados pelo heap da máquina virtual Java (JVM).
- Memory Profile: heap usage (Perfil de memória: uso do heap): a fração da memória (escala: 0 a 1) usada pelo heap da JVM.
- CPU Load (Carga da CPU): a fração da carga do sistema da CPU usada (escala: 0 a 1), exibida em porcentagem.

Visualizar métricas do Amazon CloudWatch para uma execução de trabalho do Ray

Na página de detalhes de uma execução de trabalho, abaixo da seção Run details (Detalhes da execução), você pode visualizar as métricas do trabalho. O AWS Glue Studio envia métricas de trabalho para o Amazon CloudWatch para cada trabalho executado.

O AWS Glue relata as métricas ao Amazon CloudWatch a cada 30 segundos. As métricas do AWS Glue representam valores do delta a partir dos valores relatados anteriormente. Quando apropriado, os painéis de métricas agregam (somam) os valores de 30 segundos para obter um valor para o último minuto inteiro. No entanto, as métricas do Apache Spark que o AWS Glue transfere para o Amazon CloudWatch são geralmente valores absolutos que representam o estado atual no momento em que são relatadas.

> **Note**
>
> Você deve configurar sua conta para acessar o Amazon CloudWatch, conforme descrito em .

Nas trabalhos do Ray, você pode visualizar os seguintes gráficos de métrica agregados. Com elas, você pode criar um perfil do cluster e das tarefas, e pode acessar informações detalhadas sobre cada nó. Os dados de séries temporais que sustentam esses gráficos estão disponíveis no CloudWatch para análise posterior.

Perfil da tarefa: estado da tarefa

Mostra o número de tarefas do Ray no sistema. Cada ciclo de vida da tarefa tem sua própria série temporal.

Perfil da tarefa: Nome da tarefa

Mostra o número de tarefas do Ray no sistema. Somente tarefas pendentes e ativas são mostradas. Cada tipo de tarefa (por nome) recebe sua própria série temporal.

Perfil de cluster: CPUs em uso

Mostra o número de núcleos de CPU usados. Cada nó recebe sua própria série temporal. Os nós são identificados por endereços IP, que são efêmeros e usados apenas para identificação.

Perfil do cluster: uso da memória de armazenamento de objetos

Mostra o uso de memória pelo cache de objetos do Ray. Cada localização da memória (memória física, armazenada em cache no disco e distribuída no Amazon S3) tem sua própria série temporal. O armazenamento de objetos gerencia o armazenamento de dados em todos os nós do cluster. Para obter mais informações, consulte Objects na documentação do Ray.

Perfil do cluster: número de nós

Mostra o número de nós provisionados para o cluster.

Detalhe do nó: uso da CPU

Mostra a utilização da CPU em cada nó como uma porcentagem. Cada série mostra uma porcentagem agregada do uso da CPU em todos os núcleos do nó.

Detalhe do nó: uso de memória

Mostra o uso da memória em cada nó em GB. Cada série mostra a memória agregada entre todos os processos no nó, incluindo tarefas do Ray e o processo de armazenamento do Plasma. Isso não refletirá objetos armazenados em disco ou derramados no Amazon S3.

Detalhe do nó: uso de disco

Mostra o uso de disco em cada nó em GB.

Detalhe do nó: velocidade de E/S do disco

Mostra a E/S de disco em cada nó em KB/s.

Detalhe do nó: throughput de E/S da rede

Mostra a E/S de rede em cada nó em KB/s.

Detalhe do nó: uso da CPU pelo componente do Ray

Mostra o uso da CPU em frações de um núcleo. Cada componente do Ray em cada nó recebe sua própria série temporal.

Detalhe do nó: uso da memória pelo componente do Ray

Mostra o uso da memória em GiB. Cada componente do Ray em cada nó recebe sua própria série temporal.

Detectar e processar dados sigilosos

A transformação Detect PII identifica informações de identificação pessoal (PII) em sua origem dos dados. Você escolhe a entidade de PII para identificar, como deseja que os dados sejam verificados e o que fazer com a entidade de PII identificada pela transformação Detect PII.

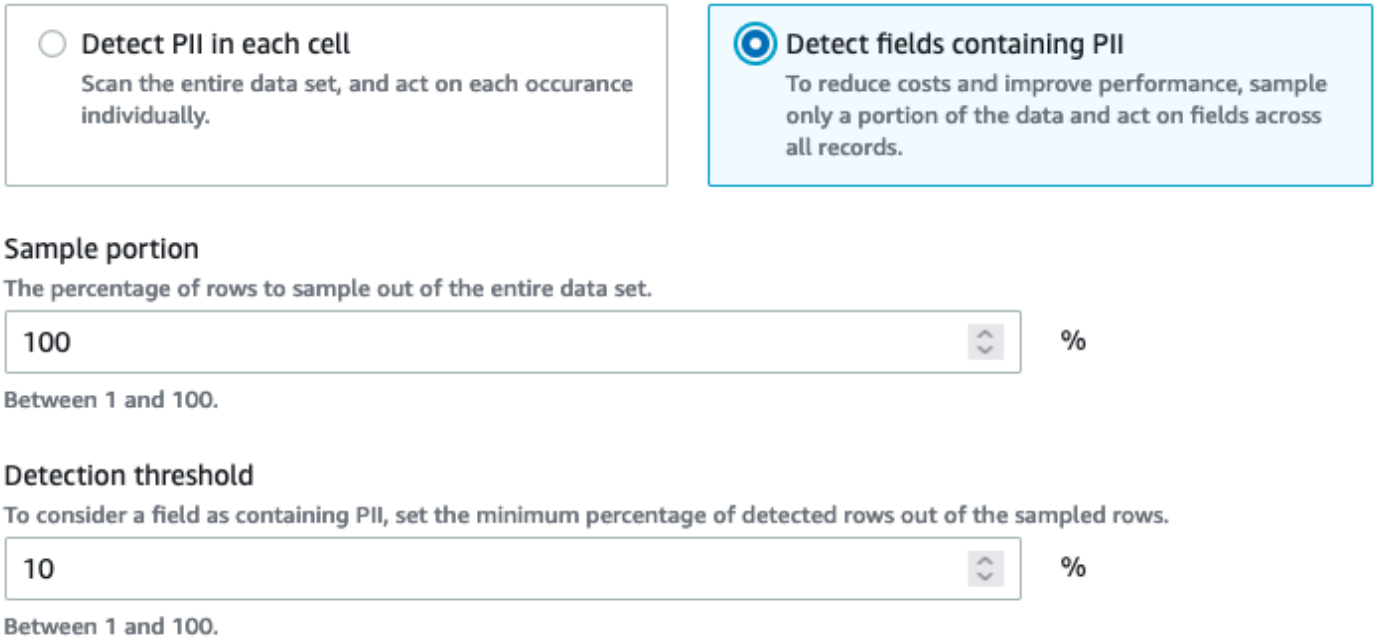
A transformação Detect PII permite detectar, mascarar ou remover entidades que você define ou que são predefinidas pela AWS. Isso permite aumentar a conformidade e reduzir a responsabilidade. Por exemplo, talvez você queira garantir que não existam informações de identificação pessoal em seus dados que possam ser lidas e queira mascarar números de previdência social com uma sequência fixa (como xxx-xx-xxxx), números de telefone ou endereços.

Para trabalhar com dados confidenciais fora do AWS Glue Studio, consulte Usando a detecção de dados confidenciais fora do AWS Glue Studio

Tópicos

Como escolher a forma como os dados serão lidos

Ao examinar seu conjunto de dados em busca de dados confidenciais, como informações de identificação pessoal (PII), é possível optar por detectar PII em cada linha ou detectar as colunas que contêm dados de PII.



Ao escolher a opção Detect PII in each cell (Detectar PII em cada célula), você está escolhendo verificar todas as linhas na origem dos dados. Esta é uma leitura abrangente para garantir que as entidades de PII sejam identificadas.

Ao escolher a opção Detect fields containing PII (Detectar campos contendo PII), você está optando por ler uma amostra de linhas para entidades de PII. Essa é uma maneira de manter os custos e os recursos baixos e, ao mesmo tempo, identificar os campos em que as entidades de PII são encontradas.

Ao optar por detectar campos que contêm PII, você pode reduzir custos e melhorar a performance por meio da amostragem de uma parte das linhas. Escolher essa opção permitirá que você especifique opções adicionais:

- Sample portion (Porção da amostra): permite especificar a porcentagem de linhas que serão amostradas. Por exemplo, ao inserir "50", estará especificando que deseja 50% das linhas lidas para a entidade PII.
- Detection threshold (Limite de detecção): permite que você especifique a porcentagem de linhas que contêm a entidade PII para que toda a coluna seja identificada como tendo a entidade PII. Por exemplo, ao digitar "10", você estará especificando que o número da entidade de PII, US Phone, nas linhas lidas deve ser 10% ou maior para que o campo seja identificado como tendo a entidade de PII US Phone. Se a porcentagem de linhas que contêm a entidade de PII for inferior a 10%, esse campo não será rotulado como tendo a entidade PII US Phone nele.

Escolha das entidades de PII para detecção

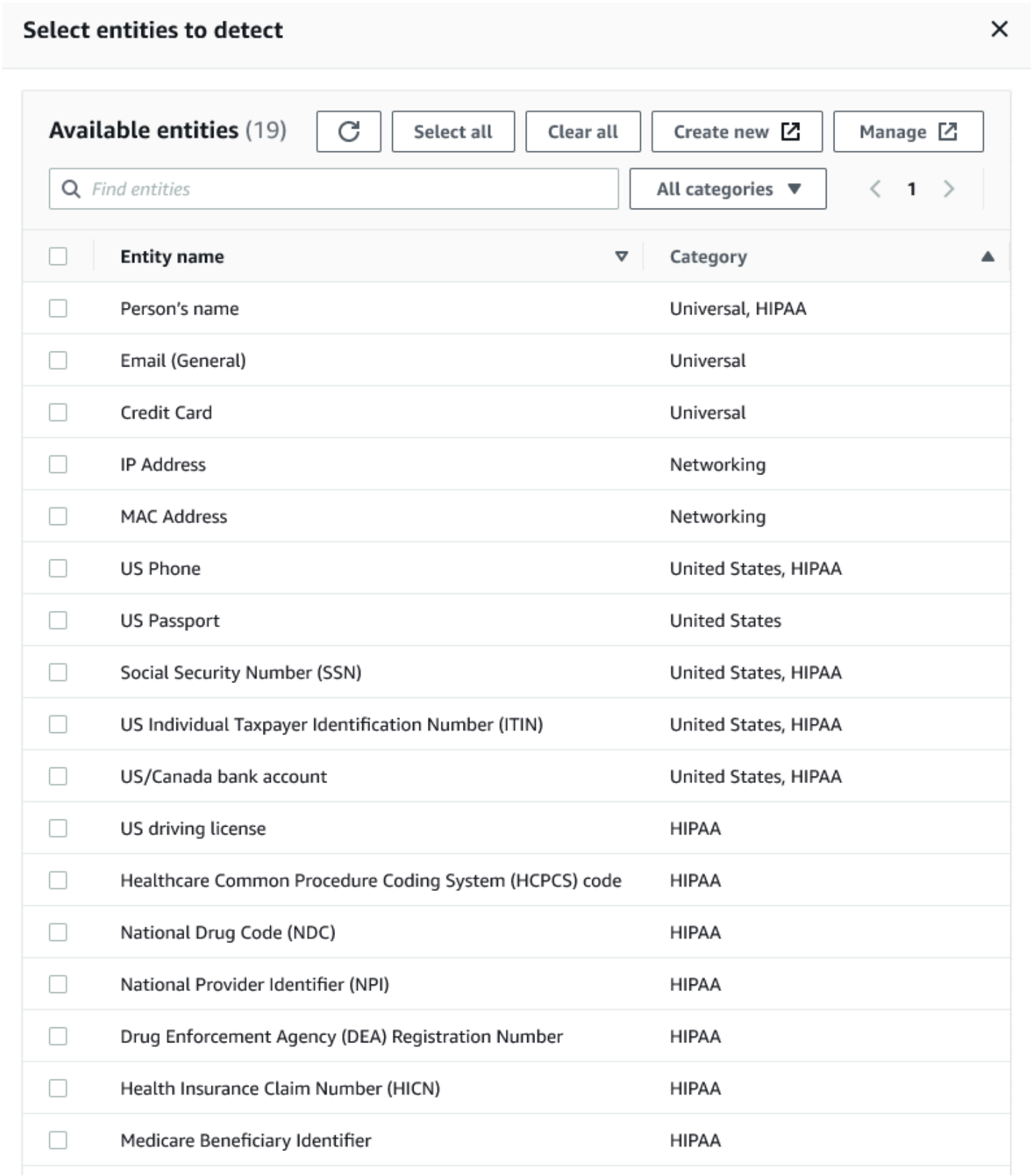
Se optou por Detect PII in each cell (Detectar PII em cada célula), escolha entre uma de três opções:

- Todos os padrões de PII disponíveis - isso inclui AWS entidades.
- Selecionar categorias: quando você selecionar categorias, os padrões de PII incluirão automaticamente os padrões nas categorias selecionadas.
- Selecionar padrões específicos: somente os padrões selecionados serão detectados.

Para obter uma lista dos tipos de dados confidenciais gerenciados, consulte Tipos de dados gerenciados.

Escolher entre todos os padrões de PII disponíveis

Se você escolher Todos os padrões de PII disponíveis, selecione entidades predefinidas por. AWSÉ possível selecionar uma, mais de uma ou todas as entidades.


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

Selecionar categorias

Se escolheu Select categories (Selecionar categorias) como os padrões de PII a serem detectados, selecione entre as opções no menu suspenso. Observe que algumas entidades podem pertencer a mais de uma categoria. Por exemplo, Person's name (Nome do indivíduo) é uma entidade que pertence às categorias Universal e HIPAA.

- Universal (exemplos: e-mail, cartão de crédito)
- HIPAA (exemplos: carteira de habilitação dos EUA, código do Healthcare Common Procedure Coding System [HCPCS – Sistema de Codificação de Procedimentos Comuns de Saúde])
- Redes (exemplos: endereço IP, endereço MAC)
- Argentina
- Austrália
- Áustria
- Bélgica
- Bósnia
- Bulgária
- Canadá
- Chile
- Colômbia
- Croácia
- Chipre
- Tchéquia
- Dinamarca
- Estônia
- Finlândia
- França
- Alemanha
- Grécia
- Hungria
- Irlanda

- Coreia
- Japão
- México
- Holanda
- Nova Zelândia
- Noruega
- Portugal
- Romênia
- Cingapura
- Eslováquia
- Eslovênia
- Espanha
- Suécia
- Suíça
- Turquia
- Ucrânia
- Estados Unidos
- Reino Unido
- Venezuela

Selecionar padrões específicos

Se escolher Select specific patterns (Selecionar padrões específicos) como os padrões de PII a serem detectados, você pode pesquisar ou navegar em uma lista de padrões que já criou, ou criar um novo padrão de entidade de detecção.

As etapas abaixo descrevem como criar um novo padrão personalizado para detectar dados sigilosos. Você criará o padrão personalizado inserindo um nome para o padrão personalizado, adicionará uma expressão regular e, opcionalmente, definirá palavras de contexto.

1. Para criar um novo padrão, clique no botão Create new (Criar novo)

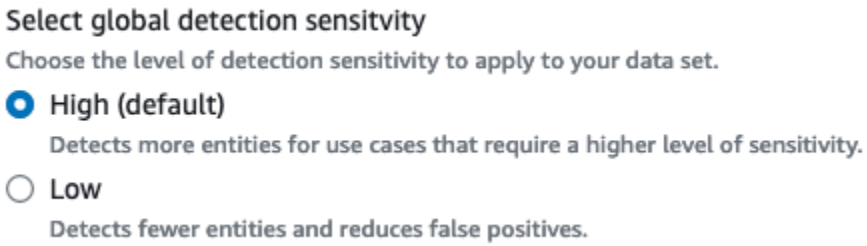
2. Na página Create detection entity (Criar entidade de detecção), insira o nome da entidade e uma expressão regular. A expressão regular (Regex) é o que o AWS Glue usará para fazer a correspondência de entidades.
3. Clique em Validate (Validar). Se a validação for bem-sucedida, você verá uma mensagem de confirmação informando que a string é uma expressão regular válida. Se a validação não for bem-sucedida, você verá uma mensagem informando que a string não está em conformidade com a formatação adequada e com os caracteres, operadores ou construções aceitos.
4. Você pode optar por adicionar palavras de contexto além da expressão regular. Palavras de contexto podem aumentar a probabilidade de uma correspondência. Elas podem ser úteis em casos nos quais os nomes de campo não descrevem a entidade. Por exemplo, os números da previdência social dos EUA podem ser nomeados “SSN” ou “SS”. A adição dessas palavras de contexto pode ajudar na correspondência da entidade.
5. Clique em Create (Criar) para criar a entidade de detecção. Qualquer entidade criada fica visível no console do AWS Glue Studio. Clique em Detection entities (Entidades de detecção) no menu de navegação à esquerda.

   Você pode editar, excluir ou criar entidades de detecção na página Detection entities (Entidades de detecção). Você também pode pesquisar por um padrão usando o campo de pesquisa.

Especificar o nível da sensibilidade de detecção

É possível definir o nível de sensibilidade ao usar a detecção de dados confidenciais.

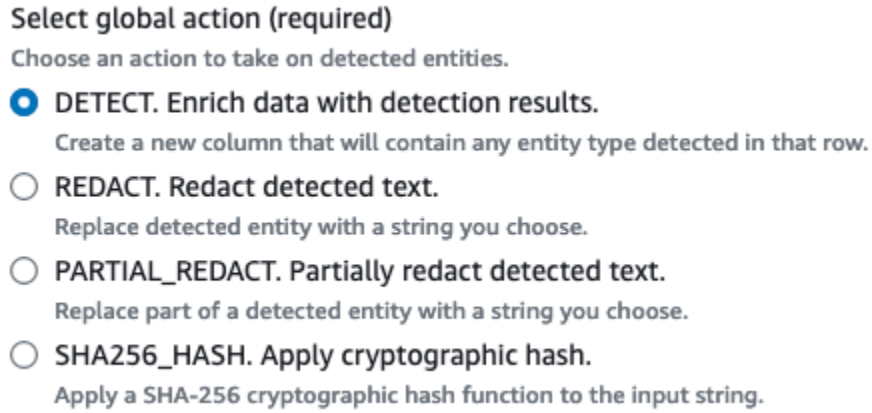
- Alto (padrão): detecta mais entidades para casos de uso que exigem um nível mais alto de sensibilidade. Todos os trabalhos do AWS Glue criados após novembro de 2023 são habilitados automaticamente com essa configuração.
- Baixo: detecta menos entidades e reduz os falsos positivos.



Como escolher o que fazer com dados de PII identificados

Se você optar por detectar PII em toda a fonte de dados, poderá selecionar uma ação global para aplicar:

- Enrich data with detection results (Enriquecer dados com resultados da detecção): se você escolher Detect PII em cada célula, poderá armazenar as entidades detectadas em uma nova coluna.
- Redact detected text (Editar o texto detectado): é possível substituir o valor de PII detectado por uma string especificada no campo opcional de entrada de texto Replacing text (Substituindo texto). Se nenhuma string for especificada, a entidade de PII detectada será substituída por "*******".
- Editar o texto detectado: é possível substituir parte das PII detectadas por uma string especificada por você. Há duas opções possíveis: deixar as extremidades desmascaradas ou mascarar fornecendo um padrão de regex explícito. Este recurso ainda não está disponível no AWS Glue 2.0.
- Aplicar hash criptográfico: você pode passar o valor de PII detectado para uma função de hash criptográfico SHA-256 e substituir o valor pela saída da função.



Diferenças entre o AWS Glue versões 2.0 e 3.0+

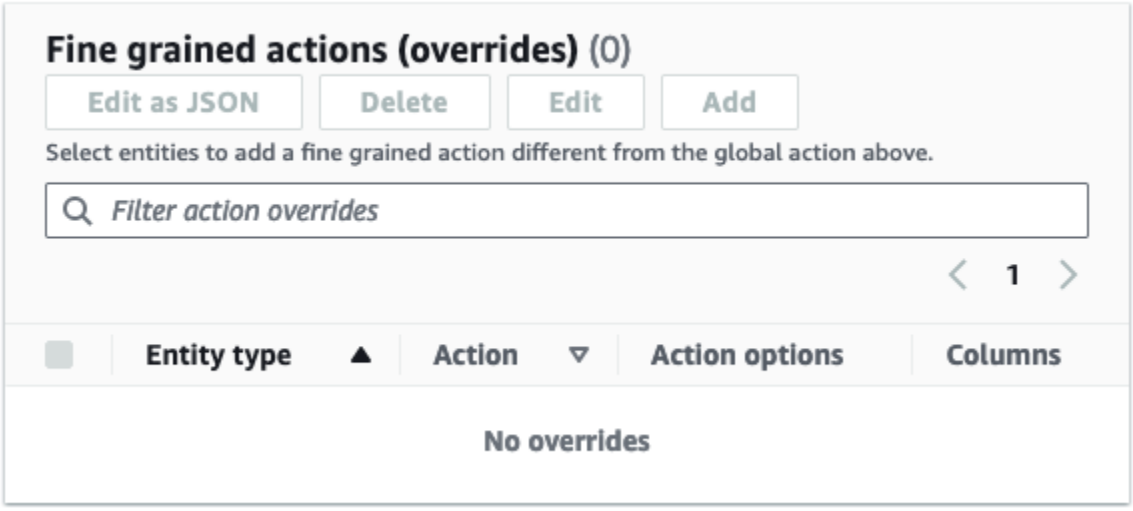
AWS GlueAs tarefas 2.0 retornarão uma nova DataFrame com as informações de PII detectadas para cada coluna em uma coluna suplementar. Qualquer redação ou trabalho de hash é visível no script AWS Glue na guia visual.

AWS GlueOs trabalhos 3.0 e 4.0 retornarão um novo DataFrame com essa mesma coluna suplementar. Uma nova chave para "actionUsed" está presente e pode ser uma de `DETECT`, `REDACT`, `PARTIAL_REDACT` ou `SHA256_HASH`. Se uma ação de mascaramento for selecionada, ela DataFrame retornará dados com dados confidenciais mascarados.

Adicionar substituições de ações refinadas

Configurações adicionais de detecção e ação podem ser adicionadas à tabela de substituições de ações refinadas. Isso permite a você:

- Incluir ou excluir determinadas colunas da detecção: um esquema inferido na fonte de dados preencherá a tabela com as colunas disponíveis.
- Especificar configurações específicas que sejam mais refinadas do que usar ações globais: por exemplo, você pode especificar diferentes configurações de redação de texto para diferentes tipos de entidade.
- Especificar uma ação diferente da ação global: se uma ação diferente precisar ser aplicada em um tipo de dados confidenciais diferente, isso pode ser feito aqui. Observe que duas edit-in-place ações diferentes (redação e hash) não podem ser usadas na mesma coluna, mas a detecção sempre pode ser usada.



Gerenciar trabalhos de ETL com o AWS Glue Studio

Você pode usar a interface gráfica simples no AWS Glue Studio para gerenciar seus trabalhos de ETL. Usando o menu de navegação, escolhar Jobs (Trabalhos) para visualizar a página Jobs (Trabalhos). Nessa página, você pode ver todos os trabalhos que criou com o AWS Glue Studio ou com o console do AWS Glue. Você pode exibir, gerenciar e executar seus trabalhos nessa página.

Você também pode executar as seguintes tarefas nessa página:

- Iniciar uma execução de trabalho
- Programar execuções de trabalho
- Gerenciar programações de trabalho
- Interromper execuções de trabalho
- Visualizar os trabalhos
- Exibir informações para execuções de trabalho recentes
- Visualizar o script de trabalho
- Modificar as propriedades do trabalho
- Salvar o trabalho
- Clonar um trabalho
- Excluir trabalhos

Iniciar uma execução de trabalho

No AWS Glue Studio, você pode executar seus trabalhos sob demanda. Um trabalho pode ser executado várias vezes, e cada vez que você o executa, o AWS Glue coleta informações sobre as atividades e a performance do trabalho. Essa informação é referida como uma execução de trabalho e é identificada por um ID de execução de trabalho.

Você pode iniciar uma execução de trabalho no AWS Glue Studio das seguintes formas:

- Na página Jobs (Trabalhos), escolha o trabalho que você deseja iniciar e, em seguida, escolha o botão Run job (Executar trabalho).
- Se você estiver visualizando um trabalho no editor visual e ele estiver salvo, você pode escolher o botão Run (Executar) para iniciar uma execução de trabalho.

Para obter mais informações sobre execuções de trabalhos, consulte Trabalhar com trabalhos no console do AWS Glue no Guia do desenvolvedor do AWS Glue.

Programar execuções de trabalho

No AWS Glue Studio, você pode criar uma programação para que seus trabalhos sejam executados em horários específicos. Você pode especificar restrições, como a quantidade de vezes que os trabalhos são executados. Essas restrições são feitas com base no `cron` e têm as mesmas limitações que o `cron`. Por exemplo, se você optar por executar seu trabalho no dia 31 de cada mês, lembre-se de que alguns meses não têm 31 dias. Para obter mais informações sobre o `cron`, consulte Expressões do Cron no Guia do desenvolvedor do AWS Glue.

Para executar trabalhos de acordo com uma programação

1. Crie uma programação de trabalho usando um dos seguintes métodos:
   - Na página Jobs (Trabalhos), escolha o trabalho para o qual você deseja criar uma programação, escolha Actions (Ações) e depois Schedule job (Programar trabalho).
   - Se você estiver visualizando um trabalho no editor visual e ele estiver salvo, escolha a guia Schedules (Programações). Em seguida, escolha Create Schedule (Criar programação).
2. Na página Schedule job run (Programar execução de trabalho), insira as seguintes informações:
   - Name (Nome): insira um nome para a programação do trabalho.
   - Frequency (Frequência): insira a frequência da programação do trabalho. Você pode escolher uma das seguintes opções:
     - Hourly (Por hora): o trabalho será executado a cada hora, começando em um minuto específico. Você pode especificar o Minute (Minuto) da hora em que o trabalho deve ser executado. Por padrão, quando você escolhe por hora, o trabalho é executado no início da hora (minuto 0).
     - Daily (Diariamente): o trabalho será executado todos os dias, iniciando em um determinado momento. Você pode especificar o Minute (Minuto) da hora em que o trabalho deve ser executado e a Start hour (Hora de início) do trabalho. As horas são especificadas usando um relógio de 23 horas, em que você usa os números 13 a 23 para as horas após meio-dia. O valor padrão para minuto e hora é zero, o que significa que se você selecionar Daily (Diariamente), por padrão, o trabalho será executado à meia-noite.
     - Weekly (Semanal): o trabalho será executado todas as semanas em um ou mais dias. Além das mesmas configurações descritas anteriormente para Daily (Diariamente), você pode

escolher os dias da semana em que o trabalho será executado. Você pode escolher um ou mais dias.

- Monthly (Mensalmente): o trabalho será executado todos os meses em um dia específico. Além das mesmas configurações descritas anteriormente para Daily (Diariamente), você pode escolher o dia do mês em que o trabalho será executado. Especifique o dia como um valor numérico de 1 a 31. Se você selecionar um dia que não existe em um mês, por exemplo, o 30º dia de fevereiro, então o trabalho não será executado nesse mês.
- Custom (Personalizado): insira uma expressão para sua programação de trabalho usando a sintaxe do `cron`. As expressões do Cron permitem que você crie programações mais complicadas, como o último dia do mês (em vez de um dia específico do mês) ou a cada terceiro mês no 7º e 21º dias do mês.

  Consulte Expressões do Cron no Guia do desenvolvedor do AWS Glue

- Description (Descrição): como opção, você pode inserir uma descrição para a programação do trabalho. Se você planeja usar a mesma programação para vários trabalhos, uma descrição facilita determinar o que a programação do trabalho faz.

3. Escolha Create schedule (Criar programação) para salvar a programação de trabalho.
4. Depois de criar a programação, uma mensagem de êxito é exibida na parte superior da página do console. Você pode escolher Job details (Detalhes do trabalho) nesse banner, para exibir os detalhes do trabalho. Isso abre a página do editor de trabalhos visual, com a guia Schedules (Programações) selecionada.

Gerenciar programações de trabalho

Depois de criar programações para um trabalho, você pode abri-lo no editor visual e escolher a guia Schedules (Programações) para gerenciar as programações.

Na guia Schedules (Programações) do editor visual, você pode executar as seguintes tarefas:

- Criar uma nova programação.

  Escolha Create schedule (Criar programação) e insira as informações da programação, conforme descrito em the section called “Programar execuções de trabalho”.

- Editar uma programação existente.

  Escolha a programação que você deseja editar e selecione Action (Ação) e, em seguida, Edit schedule (Editar programação). Quando você opta por editar uma programação existente,

Frequency (Frequência) exibe Custom (Personalizada) e a programação é exibida como uma expressão do `cron`. Você também pode modificar a expressão do `cron` ou especificar uma nova programação usando o botão Frequency (Frequência). Ao concluir as alterações, escolha Update schedule (Atualizar programação).

- Pausar uma programação ativa.

  Escolha uma programação ativa, selecione Action (Ação) e, em seguida, Pause schedule (Pausar programação). A programação é desativada instantaneamente. Escolha o botão refresh (reload) [atualizar (recarregar)] para ver o status da programação de trabalho atualizada.

- Retomar uma programação pausada.

  Escolha uma programação desativada, selecione Action (Ação) e, em seguida, resume schedule (Retomar programação). A programação é ativada instantaneamente. Escolha o botão refresh (reload) [atualizar (recarregar)] para ver o status da programação de trabalho atualizada.

- Excluir uma programação.

  Escolha a programação que você deseja remover, selecione Action (Ação) e, em seguida, Delete schedule (Excluir programação). A programação é excluída instantaneamente. Escolha o botão refresh (reload) [atualizar (recarregar)] para ver a lista de programações de trabalho atualizada. A programação terá o status Deleting (Excluíndo) até que tenha sido completamente removida.

Interromper execuções de trabalho

Você pode interromper um trabalho antes que ele tenha concluído sua execução. Você pode escolher essa opção se souber que o trabalho não está configurado corretamente ou se ele está demorando muito para ser concluído.

Na página Monitoring (Monitoramento), na lista Job runs (Execuções de trabalho), selecione o trabalho que deseja interromper, escolha Actions (Ações) e Stop run (Interromper execução).

Visualizar os trabalhos

Você pode visualizar todos os trabalhos na página Jobs (Trabalhos). Você pode acessar essa página escolhendo Jobs (Trabalhos) no painel de navegação.

Na página Jobs (Trabalhos), você poderá ver todos os trabalhos criados na sua conta. A list Your jobs (Seus trabalhos) mostra o nome do trabalho, o tipo, o status da última execução desse trabalho

e as datas em que ele foi criado e modificado pela última vez. Você pode escolher o nome de um trabalho para ver informações detalhadas sobre ele.

Você também pode usar o painel Monitoring (Monitoramento) para exibir todos os trabalhos. Você pode acessar o painel escolhendo Monitoring (Monitoramento) no painel de navegação.

Personalizar a exibição do trabalho

Você pode personalizar como os trabalhos são exibidos na seção Your jobs (Seus trabalhos) da página Jobs (Trabalhos). Além disso, você pode inserir um texto no campo de pesquisa de texto para exibir somente trabalhos com um nome que contenha esse texto.

Se você escolher o ícone de configurações

na seção Your jobs (Seus trabalhos), poderá personalizar como AWS Glue Studio exibe as informações na tabela. Você pode optar por quebrar as linhas de texto na exibição, alterar o número de trabalhos exibidos na página e especificar quais colunas serão exibidas.

Exibir informações para execuções de trabalho recentes

Um trabalho pode ser executado várias vezes à medida que novos dados são adicionados no local de origem. Cada vez que um trabalho é executado, um ID exclusivo é atribuído à execução do trabalho e as informações sobre essa execução são coletadas. É possível exibir essas informações usando os métodos a seguir:

- Escolha a guia Runs (Execuções) do editor visual para exibir as informações de execução do trabalho exibido no momento.

  Na guia Runs (Execuções), página Recent job runs (Execuções de trabalho recentes), há um cartão para cada execução de trabalho. As informações exibidas na guia Runs (Execuções) incluem:
  - ID de execução de trabalho
  - Número de tentativas para execução desse trabalho
  - Status da execução do trabalho
  - Horário de início e de término da execução do trabalho
  - O runtime do trabalho
  - Um link para os arquivos de log do trabalho
  - Um link para os arquivos de log de erros do trabalho

  - O erro retornado para trabalhos com falha
- É possível selecionar uma execução de trabalho para ver informações adicionais sobre o trabalho, incluindo:
  - Argumentos de entrada
  - Logs contínuos
  - Métricas: é possível ver visualizações de métricas básicas. Para obter ter mais informações sobre as métricas incluídas, consulte the section called “Visualizar métricas do Amazon CloudWatch para uma execução de trabalho do Spark”.
  - Interface do usuário do Spark: é possível visualizar os logs do Spark do seu trabalho na interface do usuário do Spark. Para obter mais informações sobre o uso da interface do usuário Web do spark , consulte o the section called “Monitorar com a interface do usuário do Spark”. Habilite esse recurso seguindo o procedimento em the section called “Habilitar a interface do usuário do Spark para trabalhos”.

Você pode selecionar Visualizar detalhes para ver informações semelhantes na página de detalhes da execução do trabalho. Como alternativa, você pode navegar até a página de detalhes da execução do trabalho por meio da página Monitoramento. No painel de navegação, escolha Monitoring (Monitoramento). Role para baixo até a lista Job runs (Execuções do trabalho). Escolha o trabalho e, em seguida, escolha View run details (Visualizar os detalhes da execução). O conteúdo está descrito em Visualizar os detalhes de uma execução de trabalho.

Para obter mais informações sobre os logs de trabalhos, consulte Visualizar os logs de execuções de trabalho.

Visualizar o script de trabalho

Depois de fornecer informações para todos os nós no trabalho, o AWS Glue Studio gera um script que é usado pelo trabalho para ler os dados da origem, transformá-los e gravá-los no local de destino. Se você salvar o trabalho, poderá exibir esse script a qualquer momento.

Para exibir o script gerado para seu trabalho

1. No painel de navegação, escolha Jobs (Trabalhos).
2. Na página Jobs (Trabalhos), na lista Your jobs (Seus trabalhos), escolha o nome do trabalho que você deseja revisar. Como alternativa, você pode selecionar um trabalho na lista, escolher o menu Actions (Ações) e, em seguida, escolher Edit job (Editar trabalho).

3. Na página do editor visual, escolha a guia Script na parte superior para exibir o script de trabalho.

   Se quiser editar o script de trabalho, consulte Guia de programação do AWS Glue.

Modificar as propriedades do trabalho

Os nós no diagrama de trabalho definem as ações executadas pelo trabalho, mas há várias propriedades que você também pode configurar para o trabalho. Essas propriedades determinam o ambiente em que o trabalho é executado, os recursos que ele usa, as configurações de limite, as configurações de segurança e muito mais.

Para personalizar o ambiente de execução do trabalho

1. No painel de navegação, escolha Jobs (Trabalhos).
2. Na página Jobs (Trabalhos), na lista Your jobs (Seus trabalhos), escolha o nome do trabalho que você deseja revisar.
3. Na página do editor visual, escolha a guia Job details Detalhes do trabalho) na parte superior do painel de edição do trabalho.
4. Modifique as propriedades do trabalho, conforme necessário.

   Para obter mais informações sobre as propriedades do trabalho, consulte Definir propriedades do trabalho no Guia do desenvolvedor do AWS Glue.
5. Expanda a seção Advanced properties (Propriedades avançadas), se você precisar especificar essas propriedades de trabalho adicionais:

   - Script filename (Nome de arquivo do script): o nome do arquivo que armazena o script do trabalho no Amazon S3.
   - Script path (Caminho do script): o local do Amazon S3 em que o script do trabalho é armazenado.
   - Job metrics (Métricas de trabalho): (indisponível para trabalhos de shell do Python) ativa a criação de métricas do Amazon CloudWatch quando esse trabalho é executado.
   - Continuos logging (Registro em log contínuo): (indisponível para trabalhos de shell do Python) ativa o registro em log contínuo no CloudWatch, para que os logs estejam disponíveis para visualização antes que o trabalho seja concluído.

- Spark UI (IU do Spark) e Spark UI logs path (Caminho de logs da IU do Spark): (indisponível para trabalhos de shell do Python) ativa o uso da interface do usuário do Spark para monitorar o trabalho e especifica o local para os logs da interface do usuário do Spark.
- Maximum concurrency (Simultaneidade máxima): define o número máximo de execuções simultâneas permitidas para o trabalho.
- Temporary path (Caminho temporário): o local de um diretório de trabalho no Amazon S3 onde os resultados intermediários temporários serão gravados quando o AWS Glue executar o script.
- Delay notification threshold (minutes) (Limite de notificação de atraso [minutos]): especifica um limite de atraso para o trabalho. Se o trabalho for executado por mais tempo do que o especificado pelo limite, o AWS Glue envia uma notificação de atraso para o trabalho ao CloudWatch.
- Security configuration (Configuração de segurança) e Server-side encryption (Criptografia do lado do servidor): use esses campos para escolher as opções de criptografia do trabalho.
- Use Glue Data Catalog as the Hive metastore (Usar o Glue Data Catalog como metastore do Hive): escolha essa opção se quiser usar o AWS Glue Data Catalog como uma alternativa ao Apache Hive Metastore.
- Additional network connection(Conexão de rede adicional): para uma origem dos dados em uma VPC, você pode especificar uma conexão do tipo `Network` a fim de garantir que seu trabalho acesse seus dados por meio da VPC.
- Python library path (Caminho da biblioteca do Python), Dependent jars path (Caminho de arquivos jar dependentes, não disponível para trabalhos de shell do Python) ou Referenced files path (Caminho de arquivos referenciados): use esses campos para especificar o local dos arquivos adicionais usados pelo trabalho quando ele executa o script.
- Job Parameters (Parâmetros do trabalho): você pode adicionar um conjunto de pares de chave-valor que são transmitidos como parâmetros nomeados para o script. Nas chamadas do Python para AWS Glue APIs, é melhor transmitir os parâmetros explicitamente por nome. Para obter mais informações sobre como usar parâmetros em um script de trabalho, consulte Transmitir e acessar parâmetros do Python no AWS Glue no Guia do desenvolvedor do AWS Glue.
- Tags: você pode adicionar tags ao trabalho para ajudar a organizá-las e identificá-las.

6. Depois de modificar as propriedades do trabalho, salve-o.

Armazenar arquivos de ordem aleatória do Spark no Amazon S3

Alguns trabalhos de ETL exigem leitura e combinação de informações de várias partições, por exemplo, ao usar uma transformação de união. Essa operação é referida como shuffling (ordenamento aleatório). Durante uma geração de ordem aleatória, os dados são gravados no disco e transferidos pela rede. Com o AWS Glue versão 3.0, você pode configurar o Amazon S3 como um local de armazenamento para esses arquivos. O AWS Glue fornece um gerenciador de ordenamento aleatório que grava e lê arquivos aleatórios de e para o Amazon S3. Gravar e ler arquivos de ordem aleatória do Amazon S3 é mais lento (de 5% a 20%) em comparação com o disco local (ou o Amazon EBS, que é altamente otimizado para o Amazon EC2). No entanto, o Amazon S3 fornece capacidade de armazenamento ilimitada, assim você não precisa se preocupar com erros de "`No space left on device`" ao executar seu trabalho.

Para configurar seu trabalho a fim de usar o Amazon S3 para arquivos de ordem aleatória

1. Na página Jobs (Trabalhos), na lista Your jobs (Seus trabalhos), escolha o nome do trabalho que você deseja modificar.
2. Na página do editor visual, escolha a guia Job details Detalhes do trabalho) na parte superior do painel de edição do trabalho.

   Role para baixo até a seção Job parameters (Parâmetros do trabalho).
3. Especifique os seguintes pares de chave-valor.

   - `--write-shuffle-files-to-s3` — `true`

     Esse é o parâmetro principal que configura o gerenciador de ordenamento aleatório no AWS Glue a fim de usar buckets do Amazon S3 para gravar e ler dados de ordem aleatória. Pr padrão, esse parâmetro tem um valor `false`.
   - (Opcional) `--write-shuffle-spills-to-s3`: `true`

     Esse parâmetro permite que você descarregue arquivos de despejo em buckets do Amazon S3, o que fornece resiliência adicional ao seu trabalho do Spark no AWS Glue. Isso só é necessário para grandes workloads que despejam muitos dados no disco. Pr padrão, esse parâmetro tem um valor `false`.
   - (Opcional) `--conf spark.shuffle.glue.s3ShuffleBucket`: `S3://<`*`shuffle-bucket`*`>`

Esse parâmetro especifica o bucket do Amazon S3 que deve ser usado ao gravar os arquivos de ordem aleatória. Se você não definir esse parâmetro, o local será a pasta `shuffle-data` no local especificado como caminho temporário (`--TempDir`).

> **Note**
>
> Certifique-se de que o local do bucket de ordem aleatória está na mesma região da Região da AWS na qual o trabalho é executado.
> Além disso, o serviço de ordenamento aleatório não limpa os arquivos após a conclusão da execução do trabalho, portanto, você deve configurar as políticas de ciclo de vida de armazenamento do Amazon S3 no local do bucket de ordem aleatória. Para obter mais informações, consulte Gerenciar ciclo de vida de armazenamento no Manual do usuário do Amazon S3.

Salvar o trabalho

Um aviso em vermelho informando Job has not being saved (O trabalho não foi salvo) é exibido à esquerda do botão Save (Salvar) até que você salve o trabalho.

Para salvar seu trabalho

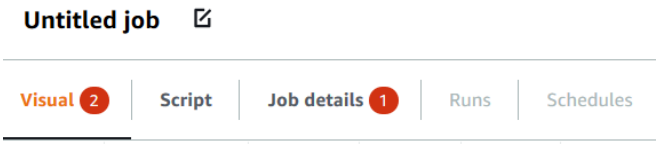
1. Forneça todas as informações necessárias nas guias Visual e Job details (Detalhes do trabalho).
2. Clique no botão Salvar.

   Depois de salvar o trabalho, o aviso “not saved” (não salvo) muda para exibir a hora e a data em que o trabalho foi salvo pela última vez.

Se você sair do AWS Glue Studio antes de salvar seu trabalho, na próxima vez que você entrar no AWS Glue Studio, uma notificação será exibida. A notificação indica que há um trabalho não salvo e pergunta se você deseja restaurá-lo. Se optar por restaurar o trabalho, poderá continuar a editá-lo.

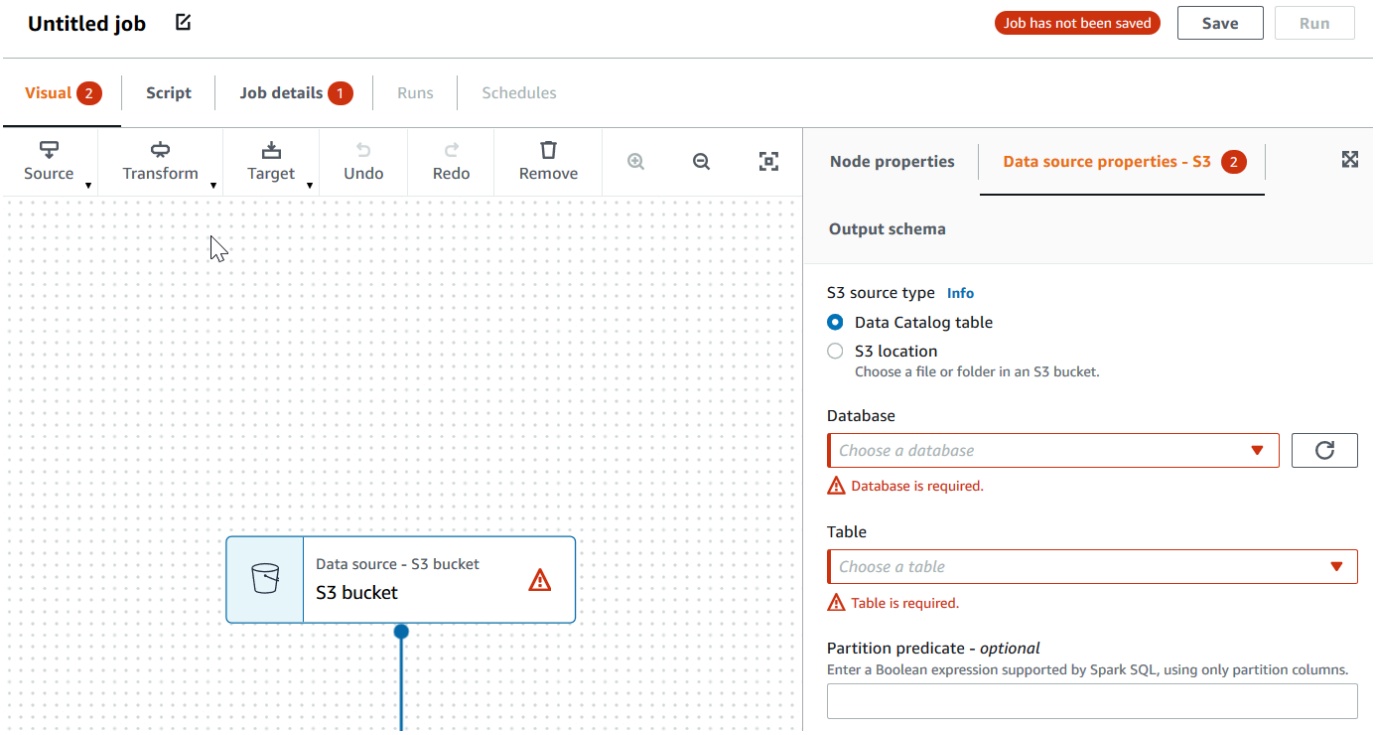
Solucionar problemas com erros ao salvar um trabalho

Se escolher o botão Save (Salvar), mas houver algumas informações necessárias faltando no trabalho, um aviso em vermelho aparecerá na guia onde as informações estão faltando. O número no aviso indica quantos campos faltantes foram detectados.



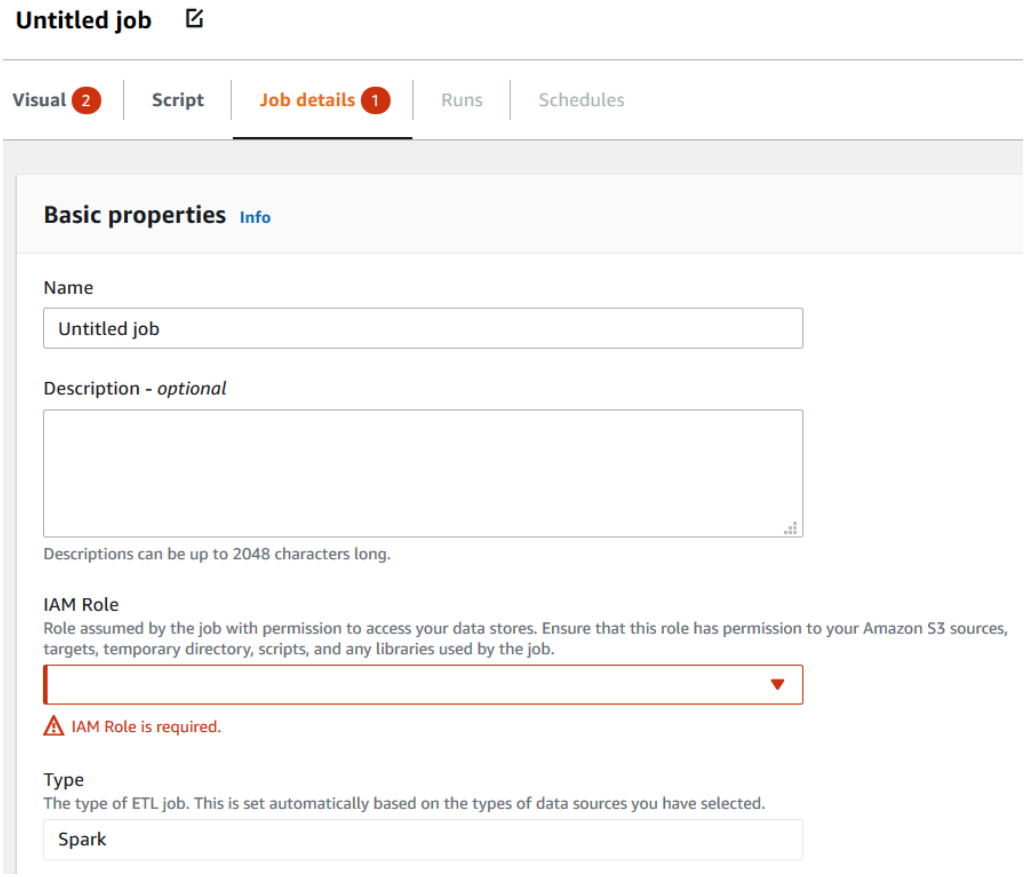
- Se um nó no editor visual não estiver configurado corretamente, a guia Visual exibirá um aviso em vermelho e o nó com o erro exibirá um símbolo de aviso ⚠.

  1. Escolher o nó. No painel de detalhes do nó, um aviso em vermelho aparece na guia onde as informações ausentes ou incorretas estão localizadas.
  2. Escolha a guia no painel de detalhes do nó que mostra um aviso em vermelho e localize os campos problemáticos, que estarão destacados. Uma mensagem de erro abaixo dos campos fornece informações adicionais sobre o problema.



- Se houver um problema com as propriedades do trabalho, a guia Job details (Detalhes do trabalho) mostrará um aviso em vermelho. Escolha essa guia e localize os campos problemáticos,

que estarão destacados. As mensagens de erro abaixo dos campos fornecem informações adicionais sobre o problema.



Clonar um trabalho

Você pode usar a ação Clone job (Clonar trabalho) para copiar um trabalho existente em um novo trabalho.

Para criar um trabalho por cópia de um trabalho existente

1. Na página Jobs (Trabalhos), na lista Your jobs (Seus trabalhos), escolha o nome do trabalho que você deseja duplicar.
2. No menu Actions (Ações), escolha Clone job (Clonar trabalho).
3. Insira um nome para o novo trabalho. Em seguida, você poderá salvar ou editar o trabalho.

Excluir trabalhos

É possível remover trabalhos que não sejam mais necessários. É possível excluir um ou mais trabalhos em uma única operação.

Para remover trabalhos do AWS Glue Studio

1. Na página Jobs (Trabalhos), na lista Your jobs (Seus trabalhos), escolha o nome do trabalho que você deseja excluir.
2. No menu Actions (Ações), escolha Delete job (Excluir trabalho).
3. Confirme que você deseja excluir o trabalho digitando **`delete`**.

Você também pode excluir um trabalho salvo ao visualizar a guia Job details (Detalhes do trabalho) desse trabalho no editor visual.

Executar trabalhos no AWS Glue

As seções a seguir fornecem mais informações sobre trabalhos de ETL e do Ray no AWS Glue.

Tópicos

Versões do AWS Glue

Você pode configurar o parâmetro de versão do AWS Glue ao adicionar ou atualizar um trabalho. A versão AWS Glue determina as versões do Apache Spark e do Python compatíveis com o AWS Glue. A versão do Python indica a versão compatível com trabalhos do tipo Spark. A tabela a seguir lista as versões disponíveis do AWS Glue, as versões correspondentes do Spark e do Python e outras alterações na funcionalidade.

Versões do AWS Glue

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
AWS Glue4.0	Versões do ambiente Spark • Spark 3.3.0 • Python 3.10	Java 8	O AWS Glue 4.0 é versão mais recente do AWS Glue. Há várias otimizações e atualizações incorporadas a essa versão do AWS Glue, como:

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade

			• Muitas atualizações de funcionalidade do Spark 3.1 para o Spark 3.3:
			  • Várias melhorias de funcionalidade quando emparelhado com o pandas. Para obter mais informações, consulte What's New in Spark 3.3 (O que há de novo no Spark 3.3).
			  • Otimizações adicionais desenvolvidas no Amazon EMR.
			  • Atualização para o EMR File System (EMRFS) 2.53.
			• Migração do Log4j 1.x para o Log4j 2
			• Várias atualizações do módulo do Python a partir do AWS Glue 3.0, como uma versão atualizada do Boto.

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			• Atualização de vários conectores, incluindo o conector padrão do Amazon Redshift. Consulte Apêndice C: Atualizações de conectores. • Atualização de vários drivers JDBC. Consulte Apêndice B: upgrades do driver JDBC. • Atualizado com um novo conector do Amazon Redshift e driver JDBC. • Suporte nativo para estruturas de data lakes abertos com o Apache Hudi, o Delta Lake e o Apache Iceberg. • Suporte nativo para o Cloud Shuffle Storage Plugin baseado no Amazon S3 (um plug-in do Apache Spark) para usar o Amazon S3 para

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			capacidade de armazenamento elástico e aleatório. Limitações As limitações com o AWS Glue 4.0 são as seguintes: • As transformações de machine learning e as informações de identificação pessoal (PII) do AWS Glue ainda não estão disponíveis no AWS Glue 4.0. Para obter mais informações sobre como migrar para o AWS Glue versão 4.0, consulte Migrar trabalhos do AWS Glue for Spark para o AWS Glue versão 4.0.

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
	Versões do ambiente Ray • Ray 2.4.0 Python 3.9	N/D	Crie e execute aplicativos Python distribuídos com AWS Glue for Ray. • Suporta distribuição de dados Ray-2.4.0 (`ray[data]`) com Python 3.9. Para obter mais informações sobre essa versão do Ray, consulte Ray-2.4.0 no repositório do Ray. GitHub • Suporta a instalação de bibliotecas Python adicionais no ambiente de runtime do `Ray2.4`. Para ter mais informações, consulte the section called “Módulos adicionais do Python para trabalhos do Ray”. • Integra registros e métricas de trabalhos do Ray com a Amazon

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			CloudWatch. Para obter mais informações, consulte the section called “Solucionar problemas de erros do Ray” e the section called “Métricas de trabalho do Ray”. • Agrega e visualiza métricas para trabalhos do Ray em AWS Glue Studio, em cada página de execução do trabalho. • Oferece suporte à distribuição de arquivos para cada diretório de trabalho em seu cluster, transferindo objetos do armazenamento de objetos do Ray para o Amazon S3 e controlando o número mínimo de nós de processamento alocados para seu trabalho do Ray. Para ter

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			mais informações, consulte the section called “Parâmetros de trabalho do Ray”. Limitações de trabalhos do Ray no AWS Glue 4.0 • AWS Glue as sessões interativas para Ray permanecem em pré-visualização para esta versão. • AWS Glue A integração do for Ray com a Amazon VPC não está disponível no momento. Os recursos em uma VPC em não AWS estarão acessíveis sem uma rota pública. Para obter mais informações sobre o uso AWS Glue com a Amazon VPC, consulte. the section called

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			“Endpoint da VPC (AWS PrivateLink)” • AWS Glue for Ray está disponível no Leste dos EUA (Norte da Virgínia), Leste dos EUA (Ohio), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio) e Europa (Irlanda).

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
AWS Glue3.0	• Spark 3.1.1 • Python 3.7	Java 8	Além da atualização do mecanismo Spark para 3.0, há otimizações e atualizações incorporadas nessa versão do AWS Glue, como: • Compila a biblioteca de ETL do AWS Glue em relação ao Spark 3.0, que é uma versão importante do Spark. • Os trabalhos de transmissão são suportados no AWS Glue 3.0. • Inclui novas otimizações do runtime do Spark no AWS Glue para performance e confiabilidade: • Processamento colunar na memória mais rápido baseado no Apache Arrow para leitura de dados CSV.

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			• Execução baseada em SIMD para leituras vetorizadas com dados CSV. • A atualização do Spark também inclui otimizações adicionais desenvolvidas no Amazon EMR. • O EMRFS foi atualizado de 2.38 para 2.46, permitindo novos recursos e correções de bugs para o acesso ao Amazon S3. • Atualizadas várias dependências que eram necessárias para a nova versão do Spark. Consulte Apêndice A: atualizações de dependência notáveis. • Drivers do JDBC atualizados para

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			nossas origens de dados com suporte nativo. Consulte Apêndice B: upgrades do driver JDBC. Limitações Veja a seguir as limitações do AWS Glue 3.0: • As transformações de machine learning do AWS Glue ainda não estão disponíveis no AWS Glue 3.0. • Alguns conectores do Spark personalizados não funcionam com o AWS Glue 3.0 se dependerem do Spark 2.4 e não tiverem compatibilidade com o Spark 3.1. Para obter mais informações sobre como migrar do

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			AWS Glue versão 3.0, consulte Migrar trabalhos do AWS Glue for Spark para o AWS Glue versão 3.0.

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
AWS Glue 2.0 (descontinuado, fim do suporte)	• Spark 2.4.3 • Python 3.7	N/D	Além dos recursos fornecidos no AWS Glue versão 1.0, o AWS Glue versão 2.0 também oferece: • Uma infraestrutura atualizada para executar trabalhos de ETL do Apache Spark no AWS Glue com tempos de inicialização reduzidos. • O registro em log padrão agora é em tempo real, com streams separados para drivers e executores, além de saídas e erros. • Suporte para especificação de módulos do Python adicionais ou versões diferentes no nível do trabalho.

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			Note O AWS Glue versão 2.0 difere do AWS Glue versão 1.0 para algumas dependências e versões graças a alterações de arquitetura subjacentes. Valide os trabalhos do AWS Glue antes de migrar entre as principais versões do AWS Glue. Para obter mais informações sobre os recursos e limitações do AWS Glue versão 2.0, consulte Executar trabalhos de ETL do Spark com

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			tempos de inicialização reduzidos.

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
AWS Glue 1.0 (descontinuado, fim do suporte)	• Spark 2.4.3 • Python 2.7 • Python 3.6	N/D	É possível manter marcadores de trabalho para os formatos Parquet e ORC em trabalhos de ETL do AWS Glue (usando o AWS Glue versão 1.0). Anteriormente, só era possível marcar formatos comuns de fonte do Amazon S3, como JSON, CSV, Apache Avro e XML em trabalhos de ETL do AWS Glue. Ao definir opções de formato para entradas e saídas de ETL, é possível especificar o uso do formato do leitor/gravador Apache Avro 1.8 para oferecer suporte à leitura e gravação do tipo lógico Avro (usando o AWS Glue versão 1.0). Anteriormente, somente o formato da versão de leitor/gravador Avro 1.7 era compatível.

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
			O tipo de conexão do DynamoDB é compatível com uma opção de gravador (usando o AWS Glue versão 1.0). Limitações As limitações com o AWS Glue 1.0 são as seguintes: • A partir de agora, as versões 0.9 e 1.0 do AWS Glue não estarão disponíveis na Ásia-Pacífico (Jacarta) (`ap-southeast-3`), Oriente Médio (EAU) (`me-central-1`) ou em outras regiões novas.

AWS Glue versão	Versões do ambiente de runtime compatíveis	Versão Java suportada	Alterações na funcionalidade
AWS Glue 0.9 (descontinuado, fim do suporte)	• Spark 2.2.1 • Python 2.7	N/D	Os trabalhos criados sem especificar uma versão do AWS Glue usam como padrão o AWS Glue 0.9. Limitações As limitações com o AWS Glue 0.9 são as seguintes: • A partir de agora, as versões 0.9 e 1.0 do AWS Glue não estarão disponíveis na Ásia-Pacífico (Jacarta) (`ap-southeast-3`), Oriente Médio (EAU) (`me-central-1`) ou em outras regiões novas.

Executar trabalhos de ETL do Spark com tempos de inicialização reduzidos

O AWS Glue versão 2.0 ou posterior fornece uma infraestrutura atualizada para executar trabalhos de ETL (extração, transformação e carregamento) do Apache Spark no AWS Glue com tempos de inicialização reduzidos. Com os tempos de espera reduzidos, os engenheiros de dados podem ser mais produtivos e aumentar sua interatividade com o AWS Glue. A variação reduzida nos horários de início do trabalho pode ajudar a atender ou exceder seus SLAs de disponibilização de dados para análise.

Para usar esse recurso com seus trabalhos de ETL do AWS Glue, escolha a versão **`2.0`** ou posterior para `Glue version` ao criar seus trabalhos.

Tópicos

Novos recursos compatíveis

Esta seção descreve os novos recursos com suporte pelo AWS Glue versão 2.0 e posteriores.

Compatível com especificação de módulos do Python adicionais no nível do trabalho

O AWS Glue versão 2.0 e posteriores também permite que você forneça módulos do Python adicionais ou versões diferentes no nível do trabalho. Você pode usar a opção `--additional-python-modules` com uma lista de módulos do Python separados por vírgulas para adicionar um novo módulo ou alterar a versão de um existente.

Por exemplo, para atualizar ou adicionar um novo módulo `scikit-learn`, use a seguinte chave-valor: `"--additional-python-modules", "scikit-learn==0.21.3"`.

Além disso, dentro da opção `--additional-python-modules`, você pode especificar um caminho do Amazon S3 para um módulo wheel do Python. Por exemplo:

```
--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
```

O AWS Glue usa o Python Package Installer (pip3) para instalar os módulos adicionais. Você pode passar opções adicionais especificadas pelo `python-modules-installer-option` para o pip3, para instalar os módulos. Qualquer incompatibilidade ou limitação do pip3 será aplicada.

Módulos do Python já fornecidos no AWS Glue versão 2.0

O AWS Glue versão 2.0 já vem com suporte aos seguintes módulos do Python:

- 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
- s3transferi—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
- s3fs—0.4.0
- Cython—0.29.15
- joblib—0.14.1
- pmdarima—1.5.3
- scikit-learn—0.22.1
- tbats—1.0.9

Comportamento de registro em log

O AWS Glue versão 2.0 e posteriores oferece suporte a um comportamento de registro em log padrão diferente. As diferenças incluem:

- O registro ocorre em tempo real.
- Existem transmissões separadas para drivers e executores.
- Para cada driver e executor, existem duas transmissões: a de saída e a de erro.

Fluxos de driver e executor

As transmissões de driver são identificadas pelo ID de execução do trabalho. As transmissões de executor são identificadas pelo trabalho <*id de execução*>_<*id da tarefa do executor*>. Por exemplo:

- `"logStreamName": "jr_8255308b426fff1b4e09e00e0bd5612b1b4ec848d7884cebe61ed33a31789..._g-f65f617bd31d54bd94482af755b6cdf464542..."`

Fluxos de saída e de erros

A transmissão de saída tem a saída padrão (stdout) do seu código. A transmissão de erros tem mensagens de registro em log do seu código/biblioteca.

- Transmissões de log:

- As transmissões de log de driver têm <*jr*>, em que <*jr*> é o ID da execução do trabalho.
- As transmissões de log de executor têm <*jr*>_<*g*>, em que <*g*> é o ID da tarefa para o executor. Você pode procurar o ID da tarefa do executor no log de erros do driver.

Os grupos de logs padrão para o AWS Glue versão 2.0 são os seguintes:

- `/aws-glue/jobs/logs/output` para a saída
- `/aws-glue/jobs/logs/error` para erros

Quando uma configuração de segurança é fornecida, os nomes de grupos de logs mudam para:

- `/aws-glue/jobs/<security configuration>-role/<Role Name>/output`
- `/aws-glue/jobs/<security configuration>-role/<Role Name>/error`

No console, o link Logs aponta para o grupo de logs de saída e o link Error (Erro) aponta para o grupo de logs de erros. Quando o registro em log contínuo está habilitado, os links Logs apontam para o grupo de logs contínuo e o link Output (Saída) aponta para o grupo de logs de saída.

Regras de registro em log

> **Note**
>
> O nome do grupo de logs padrão para registro em log contínuo é `/aws-glue/jobs/logs-v2`.

No AWS Glue versão 2.0 e posteriores, o registro em log contínuo tem o mesmo comportamento que no AWS Glue versão 1.0:

- Grupo de logs padrão: `/aws-glue/jobs/logs-v2`
- Transmissão de log do driver: <*jr*>-driver
- Transmissão de log do executor: <*jr*>-<*ID do executor*>

  O nome do grupo de logs pode ser alterado ao definir `--continuous-log-logGroupName`

  O nome das transmissões de log pode ser prefixado definindo `--continous-log-logStreamPrefix`

Recursos não compatíveis

Os seguintes recursos do AWS Glue não são suportados:

- Endpoints de desenvolvimento
- O AWS Glue versão 2.0 e posteriores não é executado no Apache YARN, portanto, as configurações do YARN não se aplicam
- O AWS Glue versão 2.0 e posteriores não tem um Hadoop Distributed File System (HDFS)
- O AWS Glue versão 2.0 e posteriores não usa alocação dinâmica, portanto, as métricas ExecutorAllocationManager não estão disponíveis
- Para trabalhos do AWS Glue versão 2.0 e posteriores, especifique o número de operadores e o tipo de operador, mas não especifique uma `maxCapacity`.
- O AWS Glue versão 2.0 e posteriores não suporta `s3n` fora da caixa. Recomendamos usar `s3` ou `s3a`. Se os trabalhos precisarem usar `s3n` por qualquer motivo, você poderá usar o seguinte argumento adicional:

```
--conf spark.hadoop.fs.s3n.impl=com.amazon.ws.emr.hadoop.fs.EmrFileSystem
```

Migrar trabalhos do AWS Glue for Spark para o AWS Glue versão 3.0

Este tópico descreve as alterações entre as versões 0.9, 1.0, 2.0 e 3.0 do AWS Glue, para permitir que você migre suas aplicações do Spark e trabalhos de ETL para o AWS Glue 3.0.

Para usar esse recurso com seus trabalhos de ETL do AWS Glue, escolha **`3.0`** para a `Glue version` ao criar seus trabalhos.

Tópicos

- Apêndice B: upgrades do driver JDBC

Novos recursos compatíveis

Esta seção descreve novos recursos e vantagens do AWS Glue versão 3.0.

- Ele é baseado no Apache Spark 3.1.1, que tem otimizações do Spark de código aberto e foi desenvolvido pelo AWS Glue e serviços de EMR, como execução de consultas adaptáveis, leitores vetorizados e agrupamento de partições e ordem aleatória otimizados.
- Drivers JDBC atualizados para todas as fontes nativas do Glue, incluindo MySQL, Microsoft SQL Server, Oracle, PostgreSQL, MongoDB e bibliotecas e dependências do Spark atualizadas trazidas pelo Spark 3.1.1.
- Acesso otimizado ao Amazon S3 com EMRFS atualizado e committers de saída otimizados do Amazon S3 habilitados por padrão.
- Acesso otimizado ao Data Catalog com índices de partição, predicados push down, listagem de partições e cliente metastore do Hive atualizado.
- Integração com Lake Formation para tabelas de catálogo controladas com filtragem em nível de célula e transações de data lake.
- Experiência de interface do usuário do Spark aprimorada com o Spark 3.1.1, com novas métricas de memória do executor do Spark e métricas de transmissão estruturadas do Spark.
- Latência de inicialização reduzida, melhorando os tempos de conclusão de trabalho e a interatividade em geral, semelhante ao AWS Glue 2.0.
- Os trabalhos do Spark são cobrados em incrementos de um segundo com uma duração mínima de faturamento dez vezes mais baixa (de um mínimo de dez minutos a um mínimo de um minuto), semelhante ao AWS Glue 2.0.

Ações para migrar para o AWS Glue 3.0

Para trabalhos existentes, altere a `Glue version` da versão anterior para `Glue 3.0` na configuração do trabalho.

- No console, escolha `Spark 3.1, Python 3 (Glue Version 3.0) or Spark 3.1, Scala 2 (Glue Version 3.0)` em `Glue version`.
- No AWS Glue Studio, escolha `Glue 3.0 - Supports spark 3.1, Scala 2, Python 3` em `Glue version`.

- Na API, escolha **`3.0`** no parâmetro `GlueVersion` da API `UpdateJob`.

Para novos trabalhos, escolha `Glue 3.0` ao criar um trabalho.

- No console, escolha `Spark 3.1, Python 3 (Glue Version 3.0) or Spark 3.1, Scala 2 (Glue Version 3.0)` em `Glue version`.
- No AWS Glue Studio, escolha `Glue 3.0 - Supports spark 3.1, Scala 2, Python 3` em `Glue version`.
- Na API, escolha **`3.0`** no parâmetro `GlueVersion` da API `CreateJob`.

Para visualizar os logs de eventos do Spark no AWS Glue 3.0, inicie um servidor de histórico do Spark atualizado para o Glue 3.0 usando o CloudFormation ou o Docker.

Lista de verificação de migração

Revise esta lista de verificação para migração.

- O seu trabalho depende do HDFS? Se sim, tente substituir o HDFS pelo S3.
  - Pesquise o caminho do sistema de arquivos começando com `hdfs://` ou `/` como o caminho DFS no código de script de trabalho.
  - Confira se o sistema de arquivos padrão não está configurado com HDFS. Se ele estiver configurado explicitamente, você precisará remover a configuração `fs.defaultFS`.
  - Confira se o seu trabalho contém algum parâmetro `dfs.*`. Se ele contiver algum, você precisará verificar se está não há problema em desabilitar os parâmetros.
- Seu trabalho depende do YARN? Se sim, verifique os impactos conferindo se o trabalho contém os seguintes parâmetros. Se ele contiver algum, você precisará verificar se está não há problema em desabilitar os parâmetros.
  - `spark.yarn.*`

    Por exemplo:

    ```
    spark.yarn.executor.memoryOverhead
    spark.yarn.driver.memoryOverhead
    spark.yarn.scheduler.reporterThread.maxFailures
    ```

  - `yarn.*`

Por exemplo:

```
yarn.scheduler.maximum-allocation-mb
yarn.nodemanager.resource.memory-mb
```

- O seu trabalho depende do Spark 2.2.1 ou do Spark 2.4.3? Se sim, verifique os impactos conferindo se o trabalho usa recursos alterados no Spark 3.1.1.
  - `https://spark.apache.org/docs/latest/sql-migration-guide.html#upgrading-from-spark-sql-22-to-23`

    Por exemplo a função `percentile_approx`, ou `SparkSession` com `SparkSession.builder.getOrCreate()` quando já existe um `SparkContext`.
  - `https://spark.apache.org/docs/latest/sql-migration-guide.html#upgrading-from-spark-sql-23-to-24`

    Por exemplo, a função `array_contains`, ou as funções `CURRENT_DATE` ou `CURRENT_TIMESTAMP` com `spark.sql.caseSensitive=true`.
- Os jars extras do seu trabalho entram em conflito no Glue 3.0?
  - Do AWS Glue 0.9/1.0: jars extras fornecidos em trabalhos do AWS Glue 0.9/1.0 existentes podem trazer conflitos de classpath devido à dependências novas ou atualizadas disponíveis no Glue 3.0. Você pode evitar conflitos de classpath no AWS Glue 3.0 com o parâmetro de trabalho `--user-jars-first` do AWS Glue ou usar shade em suas dependências.
  - Do AWS Glue 2.0: você ainda pode evitar conflitos de classpath no AWS Glue 3.0 com o parâmetro de trabalho `--user-jars-first` do AWS Glue ou usar shade em suas dependências.
- Seus trabalhos dependem do Scala 2.11?
  - O AWS Glue 3.0 usa o Scala 2.12, por isso é preciso reconstruir suas bibliotecas com o Scala 2.12, se suas bibliotecas dependem do Scala 2.11.
- As bibliotecas externas de Python do seu trabalho dependem do Python 2.7/3.6?
  - Use os parâmetros `--additional-python-modules` em vez de definir o arquivo egg/wheel/zip no caminho da biblioteca do Python.
  - Atualize as bibliotecas dependentes do Python 2.7/3.6 para Python 3.7, pois o Spark 3.1.1 removeu o suporte ao Python 2.7.

Migração do AWS Glue 0.9 para o AWS Glue 3.0

Observe as seguintes alterações ao migrar:

- O AWS Glue 0.9 usa o Spark 2.2.1 de código aberto e o AWS Glue 3.0 usa o Spark 3.1.1 otimizado para EMR.
  - Várias alterações do Spark sozinhas podem exigir a revisão de seus scripts para garantir que os recursos removidos não estejam sendo referenciados.
  - Por exemplo, o Spark 3.1.1 não habilita UDFs sem tipo Scala, mas o Spark 2.2 as permite.
- Todos os trabalhos no AWS Glue 3.0 serão executados com tempos de inicialização significativamente melhorados. Os trabalhos do Spark serão cobrados em incrementos de um segundo com uma duração mínima de cobrança dez vezes menor, já que a latência de inicialização passará de um máximo de dez minutos para um minuto.
- O comportamento de registro em log foi alterado desde o AWS Glue 2.0.
- Várias atualizações de dependência, destacadas em Apêndice A: atualizações de dependência notáveis.
- Scala também foram atualizadas para o 2.12 a partir do 2.11, e o Scala 2.12 não é compatível com o Scala 2.11.
- O Python 3.7 também é a versão padrão usada para scripts de Python, pois o AWS Glue 0.9 utilizava apenas o Python 2.
  - O Python 2.7 não é suportado com o Spark 3.1.1.
  - Um novo mecanismo de instalação de módulos adicionais do Python está disponível.
- O AWS Glue 3.0 não é executado no Apache YARN, portanto, as configurações do YARN não se aplicam.
- O AWS Glue 3.0 não tem um Hadoop Distributed File System (HDFS).
- Quaisquer jars extras fornecidos em trabalhos do AWS Glue 0.9 podem trazer dependências conflitantes, uma vez que houve atualizações em várias dependências no 3.0 a partir do 0.9. Você pode evitar conflitos de classpath no AWS Glue 3.0 com o parâmetro de trabalho `--user-jars-first` do AWS Glue.
- O AWS Glue 3.0 ainda não oferece suporte à alocação dinâmica, portanto, as métricas ExecutorAllocationManager não estão disponíveis.
- Em trabalhos do AWS Glue 3.0, você especifica o número de operadores e o tipo de operador, mas não especifica uma `maxCapacity`.
- O AWS Glue 3.0 não suporta transformações de machine learning.

- O AWS Glue 3.0 ainda não suporta endpoints de desenvolvimento.

Consulte a documentação de migração do Spark:

- consulte Upgrading from Spark SQL 2.2 to 2.3 (Atualizando do Spark SQL 2.2 para 2.3)
- consulte Upgrading from Spark SQL 2.3 to 2.4 (Atualizando do Spark SQL 2.3 para 2.4)
- consulte Upgrading from Spark SQL 2.4 to 3.0 (Atualizando do Spark SQL 2.4 para 3.0)
- consulte Upgrading from Spark SQL 3.0 to 3.1 (Atualizando do Spark SQL 3.0 para 3.1)
- consulte Changes in Datetime behavior to be expected since Spark 3.0 (Alterações no comportamento de data e hora a serem esperadas desde o Spark 3.0).

Migração do AWS Glue 1.0 para o AWS Glue 3.0

Observe as seguintes alterações ao migrar:

- O AWS Glue 1.0 usa o Spark 2.4 de código aberto e o AWS Glue 3.0 usa o Spark 3.1.1 otimizado para EMR.
  - Várias alterações do Spark sozinhas podem exigir a revisão de seus scripts para garantir que os recursos removidos não estejam sendo referenciados.
  - Por exemplo, o Spark 3.1.1 não habilita UDFs sem tipo Scala, mas o Spark 2.4 as permite.
- Todos os trabalhos no AWS Glue 3.0 serão executados com tempos de inicialização significativamente melhorados. Os trabalhos do Spark serão cobrados em incrementos de um segundo com uma duração mínima de cobrança dez vezes menor, já que a latência de inicialização passará de um máximo de dez minutos para um minuto.
- O comportamento de registro em log foi alterado desde o AWS Glue 2.0.
- Várias atualizações de dependência, destacadas no
- Scala também foram atualizadas para o 2.12 a partir do 2.11, e o Scala 2.12 não é compatível com o Scala 2.11.
- O Python 3.7 também é a versão padrão usada para scripts de Python, pois o AWS Glue 0.9 utilizava apenas o Python 2.
  - O Python 2.7 não é suportado com o Spark 3.1.1.
  - Um novo mecanismo de instalação de módulos adicionais do Python está disponível.
- O AWS Glue 3.0 não é executado no Apache YARN, portanto, as configurações do YARN não se aplicam.

- O AWS Glue 3.0 não tem um Hadoop Distributed File System (HDFS).
- Quaisquer jars extras fornecidos em trabalhos do AWS Glue 1.0 podem trazer dependências conflitantes, uma vez que houve atualizações em várias dependências no 3.0 a partir do 1.0. Você pode evitar conflitos de classpath no AWS Glue 3.0 com o parâmetro de trabalho `--user-jars-first` do AWS Glue.
- O AWS Glue 3.0 ainda não oferece suporte à alocação dinâmica, portanto, as métricas ExecutorAllocationManager não estão disponíveis.
- Em trabalhos do AWS Glue 3.0, você especifica o número de operadores e o tipo de operador, mas não especifica uma `maxCapacity`.
- O AWS Glue 3.0 não suporta transformações de machine learning.
- O AWS Glue 3.0 ainda não suporta endpoints de desenvolvimento.

Consulte a documentação de migração do Spark:

- consulte Upgrading from Spark SQL 2.4 to 3.0 (Atualizando do Spark SQL 2.4 para 3.0)
- consulte Changes in Datetime behavior to be expected since Spark 3.0 (Alterações no comportamento de data e hora a serem esperadas desde o Spark 3.0).

Migrar do AWS Glue 2.0 para o AWS Glue 3.0

Observe as seguintes alterações ao migrar:

- Todos os parâmetros de trabalho e principais recursos existentes no AWS Glue 2.0 existirão no AWS Glue3.0.
  - O committer do S3 otimizado para EMRFS para gravar dados do Parquet no Amazon S3 é habilitado por padrão no AWS Glue 3.0. No entanto, você ainda pode desabilitar isso definindo `--enable-s3-parquet-optimized-committer` como `false`.
- O AWS Glue 2.0 usa o Spark 2.4 de código aberto, e o AWS Glue 3.0 usa o Spark 3.1.1 otimizado para EMR.
  - Várias alterações do Spark sozinhas podem exigir a revisão de seus scripts para garantir que os recursos removidos não estejam sendo referenciados.
  - Por exemplo, o Spark 3.1.1 não habilita UDFs sem tipo Scala, mas o Spark 2.4 as permite.
- O AWS Glue 3.0 também apresenta uma atualização para o EMRFS, drivers JDBC atualizados e inclusões de otimizações adicionais no próprio Spark fornecidas pelo AWS Glue.

- Todos os trabalhos no AWS Glue 3.0 serão executados com tempos de inicialização significativamente melhorados. Os trabalhos do Spark serão cobrados em incrementos de um segundo com uma duração mínima de cobrança dez vezes menor, já que a latência de inicialização passará de um máximo de dez minutos para um minuto.
- O Python 2.7 não é suportado com o Spark 3.1.1.
- Várias atualizações de dependência, destacadas em Apêndice A: atualizações de dependência notáveis.
- Scala também foram atualizadas para o 2.12 a partir do 2.11, e o Scala 2.12 não é compatível com o Scala 2.11.
- Quaisquer jars extras fornecidos em trabalhos do AWS Glue 2.0 podem trazer dependências conflitantes, uma vez que houve atualizações em várias dependências no 3.0 a partir do 2.0. Você pode evitar conflitos de classpath no AWS Glue 3.0 com o parâmetro de trabalho `--user-jars-first` do AWS Glue.
- O AWS Glue 3.0 tem paralelismo de tarefas Spark diferente para configuração de driver/executor em comparação com o AWS Glue 2.0, melhora a performance e utiliza melhor os recursos disponíveis. Tanto `spark.driver.cores` quanto `spark.executor.cores` são configurados para o número de núcleos no AWS Glue 3.0 (4 no padrão e operador `G.1X`, e 8 no operador `G.2X`). Essas configurações não alteram o tipo de trabalho ou o hardware do trabalho do AWS Glue. Você pode usar essas configurações para calcular o número de partições ou divisões para corresponder ao paralelismo da tarefa do Spark na sua aplicação do Spark.

  Em geral, os trabalhos terão uma performance semelhante ou melhor em comparação com o AWS Glue 2.0. Se os trabalhos forem executados mais lentamente, você poderá aumentar o paralelismo das tarefas passando o seguinte argumento de trabalho:

  - key: `--executor-cores` value: <*número desejado de tarefas que podem ser executadas em paralelo*>
  - O valor não deve exceder 2 vezes o número de vCPUs no tipo de operador, que é 8 no `G.1X`, 16 no `G.2X`, 32 no `G.4X` e 64 no `G.8X`. Você deve ter cuidado ao atualizar essa configuração, pois isso pode afetar a performance do trabalho, pois o aumento do paralelismo gera pressão na memória e no disco, além aplicar o controle de utilização nos sistemas de origem e de destino.
- O AWS Glue 3.0 usa o Spark 3.1, que altera o comportamento para carregamento/salvamento de carimbos de data/hora de/para arquivos parquet. Para obter mais detalhes, consulte Atualização do Spark SQL 3.0 para 3.1.

Recomendamos definir os seguintes parâmetros ao ler/escrever dados de parquet que contenham colunas de carimbo de data/hora. A configuração desses parâmetros pode resolver o problema de incompatibilidade do calendário que ocorre durante a atualização do Spark 2 para o Spark 3, para o Dynamic Frame do AWS Glue e o Spark Data Frame. Use a opção CORRECTED (CORRIGIDO) para ler o valor de data e hora como ele é; e a opção LEGACY (HERDADO) para rebasear os valores de data e hora em relação à diferença de calendário durante a leitura.

```
- 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 a documentação de migração do Spark:

- consulte Upgrading from Spark SQL 2.4 to 3.0 (Atualizando do Spark SQL 2.4 para 3.0)
- consulte Changes in Datetime behavior to be expected since Spark 3.0 (Alterações no comportamento de data e hora a serem esperadas desde o Spark 3.0).

Apêndice A: atualizações de dependência notáveis

Veja a seguir as atualizações de dependência:

Dependência	Versão no AWS Glue 0.9	Versão no AWS Glue 1.0	Versão no AWS Glue 2.0	Versão no 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

Dependência	Versão no AWS Glue 0.9	Versão no AWS Glue 1.0	Versão no AWS Glue 2.0	Versão no AWS Glue 3.0
Json4s	3.2.x	3.5.x	3.5.x	3.6.6
Arrow	N/D	0.10.0	0.10.0	2.0.0
Cliente do AWS Glue Catalog	N/D	N/D	1.10.0	3.0.0

Apêndice B: upgrades do driver JDBC

A seguir estão as atualizações do driver JDBC:

Driver	Versão do driver JDBC nas versões do AWS Glue anteriores	Versão do driver JDBC no AWS Glue 3.0
MySQL	5.1	8.0.23
Microsoft SQL Server	6.1.0	7.0.0
Bancos de dados da Oracle	11.2	21.1
PostgreSQL	42.1.0	42.2.18
MongoDB	2.0.0	4.0.0

Migrar trabalhos do AWS Glue for Spark para o AWS Glue versão 4.0

Este tópico descreve as alterações entre as versões 0.9, 1.0, 2.0 e 3.0 do AWS Glue, para permitir que você migre suas aplicações do Spark e trabalhos do ETL para o AWS Glue 4.0. Ele também descreve os recursos do AWS Glue 4.0 e as vantagens de usá-lo.

Para usar esse recurso com seus trabalhos de ETL do AWS Glue, escolha **`4.0`** para a `Glue version` ao criar seus trabalhos.

Tópicos

Novos recursos compatíveis

Esta seção descreve novos recursos e vantagens do AWS Glue versão 4.0.

- Ele é baseado no Apache Spark 3.3.0, mas inclui otimizações no AWS Glue e no Amazon EMR, como execuções de consultas adaptáveis, leitores vetorizados e agrupamento de partições e ordem aleatória otimizados.
- Drivers JDBC atualizados para todas as fontes nativas do AWS Glue, incluindo MySQL, Microsoft SQL Server, Oracle, PostgreSQL, MongoDB e bibliotecas e dependências do Spark atualizadas trazidas pelo Spark 3.3.0.
- Atualizado com um novo conector do Amazon Redshift e driver JDBC.
- Acesso otimizado ao Amazon S3 com o EMR File System (EMRFS) atualizado e committers de saída otimizados do Amazon S3 habilitados por padrão.
- Acesso otimizado ao Data Catalog com índices de partição, predicados de pushdown, listagem de partições e cliente de metastore do Hive atualizado.
- Integração com Lake Formation para tabelas de catálogo controladas com filtragem em nível de célula e transações de data lake.
- Latência de startup reduzida, melhorando os tempos de conclusão de trabalho e a interatividade em geral.
- Os trabalhos do Spark são cobrados em incrementos de um segundo com uma duração mínima de faturamento dez vezes mais baixa, de um mínimo de dez minutos para um mínimo de um minuto.

- Suporte nativo para estruturas de data lakes abertos com o Apache Hudi, o Delta Lake e o Apache Iceberg.
- Suporte nativo para o Cloud Shuffle Storage Plugin baseado no Amazon S3 (um plug-in do Apache Spark) para usar o Amazon S3 para capacidade de armazenamento elástico e aleatório.

Principais aprimoramentos do Spark 3.1.1 para o Spark 3.3.0

Observe os seguintes aprimoramentos:

- Filtragem de runtime em nível de linha (SPARK-32268).
- Aprimoramentos do ANSI (SPARK-38860).
- Melhorias nas mensagens de erro (SPARK-38781).
- Suporte a tipos complexos para o leitor vetorizado de Parquet (SPARK-34863).
- Suporte a metadados de arquivos ocultos para o Spark SQL (SPARK-37273).
- Fornece um compilador de perfil para UDFs do Python/Pandas (SPARK-37443).
- Apresenta Trigger.availableNow para executar consultas de streaming como Trigger.Once em vários lotes (SPARK-36533).
- Recursos mais abrangentes de pushdown do Datasource V2 (SPARK-38788).
- Migrar do log4j 1 para o log4j 2 (SPARK-37814).

Outras alterações notáveis

Observe as seguintes alterações:

- Alterações que podem causar interrupções
  - Elimina as referências a compatibilidade com o Python 3.6 em documentos e Python/docs (SPARK-36977).
  - Remove o hack de tupla nomeada substituindo o pickle integrado pelo cloudpickle (SPARK-32079).
  - Aumenta a versão mínima do pandas para 1.0.5 (SPARK-37465).

Ações para migrar para o AWS Glue 4.0

Para trabalhos existentes, altere a `Glue version` da versão anterior para `Glue 4.0` na configuração do trabalho.

- No AWS Glue Studio, escolha `Glue 4.0 - Supports Spark 3.3, Scala 2, Python 3` em `Glue version`.
- Na API, escolha **`4.0`** no parâmetro `GlueVersion` na operação da API `UpdateJob`.

Para novos trabalhos, escolha `Glue 4.0` ao criar um trabalho.

- No console, escolha `Spark 3.3, Python 3 (Glue Version 4.0) or Spark 3.3, Scala 2 (Glue Version 3.0)` em `Glue version`.
- No AWS Glue Studio, escolha `Glue 4.0 - Supports Spark 3.3, Scala 2, Python 3` em `Glue version`.
- Na API, escolha **`4.0`** no parâmetro `GlueVersion` na operação da API `CreateJob`.

Para visualizar os logs de eventos do Spark no AWS Glue 4.0, vindo do AWS Glue 2.0 ou anterior, inicie um servidor de histórico do Spark atualizado para o AWS Glue 4.0 usando o AWS CloudFormation ou o Docker.

Lista de verificação da migração

Revise esta lista de verificação para migração:

> **Note**
>
> Para itens da lista de verificação relacionados ao AWS Glue 3.0, consulte Lista de verificação de migração.

- As bibliotecas externas de Python do seu trabalho dependem do Python 2.7/3.6?
  - Atualize as bibliotecas dependentes do Python 2.7/3.6 para Python 3.10, pois o Spark 3.3.0 removeu o suporte ao Python 2.7 e 3.6.

Migrar do AWS Glue 3.0 para o AWS Glue 4.0

Observe as seguintes alterações ao migrar:

- Todos os parâmetros de trabalho e principais recursos existentes no AWS Glue 3.0 existirão no AWS Glue 4.0.

- O AWS Glue 3.0 usa o Spark 3.1.1 otimizado para Amazon EMR, e o AWS Glue 4.0 usa o Spark 3.3.0 otimizado para Amazon EMR.

  Várias alterações do Spark isoladas podem exigir a revisão de seus scripts para garantir que os recursos removidos não estejam sendo referenciados.
- O AWS Glue 4.0 também apresenta uma atualização para o EMRFS e o Hadoop. Para obter números de versão específicos, consulte Apêndice A: atualizações de dependência notáveis.
- O AWS SDK fornecido nas tarefas de ETL agora foi atualizado de 1.11 para 1.12.
- Todos os trabalhos do Python usarão o Python versão 3.10. Anteriormente, o Python 3.7 era usado no AWS Glue 3.0.

  Como resultado, alguns pymodules trazidos prontos para uso pelo AWS Glue são atualizados.
- O Log4j foi atualizado para o Log4j2.
  - Para obter informações sobre o caminho de migração do Log4j2, consulte a documentação do Log4j.
  - Você deve renomear qualquer arquivo log4j.properties personalizado como um arquivo log4j2.properties, com as propriedades do log4j2 apropriadas.
- Para migrar determinados conectores, consulte Migração de conectores e drivers JDBC para AWS Glue 4.0.
- O SDK de criptografia do AWS foi atualizado de 1.x para 2.x. Trabalhos do AWS Glue que usam configurações de segurança e trabalhos do AWS Glue dependentes da dependência do SDK de criptografia do AWS fornecida em runtime são afetados. Veja as instruções para migração de trabalhos do AWS Glue.

  Você pode atualizar com segurança um trabalho do AWS Glue 2.0/3.0 para um trabalho do AWS Glue 4.0 porque o AWS Glue 2.0/3.0 já contém a versão de ponte do SDK de criptografia do AWS.

Consulte a documentação de migração do Spark:

- Atualização do Spark SQL 3.1 para 3.2
- Atualização do Spark SQL 3.2 para 3.3

Migrar do AWS Glue 2.0 para o AWS Glue 4.0

Observe as seguintes alterações ao migrar:

> **Note**
>
> Para obter as etapas de migração relacionadas ao AWS Glue 3.0, consulte Migrar do AWS Glue 3.0 para o AWS Glue 4.0.

- Todos os parâmetros de trabalho e principais recursos existentes no AWS Glue 2.0 existirão no AWS Glue 4.0.
- O committer do S3 otimizado para EMRFS para gravar dados em Parquet no Amazon S3 é habilitado por padrão desde o AWS Glue 3.0. No entanto, você ainda pode desabilitar isso definindo `--enable-s3-parquet-optimized-committer` como `false`.
- O AWS Glue 2.0 usa o Spark 2.4 de código aberto e o AWS Glue 4.0 usa o Spark 3.3.0 otimizado para Amazon EMR.
  - Várias alterações do Spark sozinhas podem exigir a revisão de seus scripts para garantir que os recursos removidos não estejam sendo referenciados.
  - Por exemplo, o Spark 3.3.0 não habilita UDFs sem tipo Scala, mas o Spark 2.4 as permite.
- O AWS SDK fornecido nas tarefas de ETL agora foi atualizado de 1.11 para 1.12.
- O AWS Glue 4.0 também apresenta uma atualização para o EMRFS, drivers JDBC atualizados e inclusões de otimizações adicionais no próprio Spark fornecidas pelo AWS Glue.
- O Scala foi atualizado da versão 2.11 para a versão 2.12, e o Scala 2.12 não é compatível retroativamente com o Scala 2.11.
- O Python 3.10 também é a versão padrão usada para scripts de Python, pois o AWS Glue 2.0 utilizava apenas o Python 3.7 e 2.7.
  - O Python 2.7 não é compatível com o Spark 3.3.0. Qualquer trabalho que solicite Python 2 na configuração do trabalho falhará com uma IllegalArgumentException.
  - Um novo mecanismo de instalação de módulos adicionais do Python está disponível desde o AWS Glue 2.0.
- Várias atualizações de dependência, destacadas em Apêndice A: atualizações de dependência notáveis.
- Quaisquer arquivos JAR extras fornecidos em trabalhos do AWS Glue 2.0 podem trazer dependências conflitantes, uma vez que houve atualizações em várias dependências no 4.0 a partir do 2.0. Você pode evitar conflitos de classpath no AWS Glue 4.0 com o parâmetro de trabalho `--user-jars-first` do AWS Glue.

- O AWS Glue 4.0 usa o Spark 3.3. A partir do Spark 3.1, houve uma mudança no comportamento para carregamento/salvamento de carimbos de data/hora de/para arquivos parquet. Para obter mais detalhes, consulte Atualização do Spark SQL 3.0 para 3.1.

  Recomendamos definir os seguintes parâmetros ao ler/escrever dados de parquet que contenham colunas de carimbo de data/hora. A configuração desses parâmetros pode resolver o problema de incompatibilidade do calendário que ocorre durante a atualização do Spark 2 para o Spark 3, para o Dynamic Frame do AWS Glue e o Spark Data Frame. Use a opção CORRECTED (CORRIGIDO) para ler o valor de data e hora como ele é; e a opção LEGACY (HERDADO) para rebasear os valores de data e hora em relação à diferença de calendário durante a leitura.

  ```
  - 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 Migração de conectores e drivers JDBC para AWS Glue 4.0.
- O SDK de criptografia do AWS foi atualizado de 1.x para 2.x. Trabalhos do AWS Glue que usam configurações de segurança e trabalhos do AWS Glue dependentes da dependência do SDK de criptografia do AWS fornecida em runtime são afetados. Veja estas instruções para migração de trabalhos do AWS Glue:
  - Você pode atualizar com segurança um trabalho AWS Glue 2.0 para um trabalho AWS Glue 4.0 porque o AWS Glue 2.0 já contém a versão de ponte do SDK de criptografia do AWS.

Consulte a documentação de migração do Spark:

- Atualização do Spark SQL 2.4 para 3.0
- Atualização do Spark SQL 3.1 para 3.2
- Atualização do Spark SQL 3.2 para 3.3
- Alterações no comportamento de data e hora a serem esperadas desde o Spark 3.0.

Migrar do AWS Glue 1.0 para o AWS Glue 4.0

Observe as seguintes alterações ao migrar:

- O AWS Glue 1.0 usa o Spark 2.4 de código aberto e o AWS Glue 4.0 usa o Spark 3.3.0 otimizado para Amazon EMR.
  - Várias alterações do Spark sozinhas podem exigir a revisão de seus scripts para garantir que os recursos removidos não estejam sendo referenciados.
  - Por exemplo, o Spark 3.3.0 não habilita UDFs sem tipo Scala, mas o Spark 2.4 as permite.
- Todos os trabalhos no AWS Glue 4.0 serão executados com tempos de startup significativamente melhorados. Os trabalhos do Spark serão cobrados em incrementos de um segundo com uma duração mínima de cobrança dez vezes menor, já que a latência de startup passará de um máximo de dez minutos para um minuto.
- O comportamento de registro em log mudou significativamente no AWS Glue 4.0, o Spark 3.3.0 tem um requisito mínimo de Log4j2.
- Várias atualizações de dependência, destacadas no apêndice.
- O Scala também foi atualizado para o 2.12 a partir do 2.11, e o Scala 2.12 não é compatível com o Scala 2.11.
- O Python 3.10 também é a versão padrão usada para scripts de Python, pois o AWS Glue 0.9 utilizava apenas o Python 2.

  O Python 2.7 não é compatível com o Spark 3.3.0. Qualquer trabalho que solicite Python 2 na configuração do trabalho falhará com uma IllegalArgumentException.
- Um novo mecanismo de instalação de módulos adicionais do Python por meio do pip está disponível desde o AWS Glue 2.0. Para mais informações, consulte Instalação de módulos Python adicionais com pip no AWS Glue 2.0+.
- O AWS Glue 4.0 não é executado no Apache YARN, portanto, as configurações do YARN não se aplicam.
- O AWS Glue 4.0 não tem um Sistema de Arquivos Distribuído do Hadoop (HDFS).
- Quaisquer arquivos JAR extras fornecidos em trabalhos do AWS Glue 1.0 podem trazer dependências conflitantes, uma vez que houve atualizações em várias dependências no 4.0 a partir do 1.0. Nós habilitamos o AWS Glue 4.0 com os parâmetros de trabalho `--user-jars-first` AWS Glue por padrão, para evitar esse problema.
- Agora o AWS Glue 4.0 é compatível com auto scaling. Portanto, a métrica ExecutorAllocationManager estará disponível quando o auto scaling estiver ativado.
- Em trabalhos do AWS Glue 4.0, você especifica o número de operadores e o tipo de operador, mas não especifica uma `maxCapacity`.
- O AWS Glue 4.0 ainda não é compatível com transformações de machine learning.

- Para migrar determinados conectores, consulte Migração de conectores e drivers JDBC para AWS Glue 4.0.
- O SDK de criptografia do AWS foi atualizado de 1.x para 2.x. Trabalhos do AWS Glue que usam configurações de segurança e trabalhos do AWS Glue dependentes da dependência do SDK de criptografia do AWS fornecida em runtime são afetados. Veja estas instruções para migração de trabalhos do AWS Glue.
  - Você não pode migrar um trabalho de AWS Glue 0.9/1.0 para um trabalho do AWS Glue 4.0 diretamente. Isso ocorre porque, ao atualizar diretamente para a versão 2.x ou posterior e ativar todos os novos recursos imediatamente, o AWS Encryption SDK não conseguirá descriptografar o texto cifrado criptografado em versões anteriores do AWS Encryption SDK.
  - Para atualizar com segurança, primeiro recomendamos que você migre para um ttrabalho do AWS Glue 2.0/3.0 que contenha a versão ponte do AWS Encryption SDK. Execute o trabalho uma vez para utilizar a versão ponte do AWS Encryption SDK.
  - Após a conclusão, você pode migrar com segurança o trabalho do AWS Glue 2.0/3.0 para o AWS Glue 4.0.

Consulte a documentação de migração do Spark:

- Atualização do Spark SQL 2.4 para 3.0
- Atualização do Spark SQL 3.0 para 3.1
- Atualização do Spark SQL 3.1 para 3.2
- Atualização do Spark SQL 3.2 para 3.3
- Alterações no comportamento de data e hora a serem esperadas desde o Spark 3.0.

Migrar do AWS Glue 0.9 para o AWS Glue 4.0

Observe as seguintes alterações ao migrar:

- O AWS Glue 0.9 usa o Spark 2.2.1 de código aberto e o AWS Glue 4.0 usa o Spark 3.3.0 otimizado para Amazon EMR.
  - Várias alterações do Spark isoladas podem exigir a revisão de seus scripts para garantir que os recursos removidos não estejam sendo referenciados.
  - Por exemplo, o Spark 3.3.0 não habilita UDFs sem tipo Scala, mas o Spark 2.2 as permite.
- Todos os trabalhos no AWS Glue 4.0 serão executados com tempos de startup significativamente melhorados. Os trabalhos do Spark serão cobrados em incrementos de um segundo com uma

duração mínima de cobrança dez vezes menor, porque a latência de startup passará de um máximo de dez minutos para um minuto.

- O comportamento de log mudou significativamente desde o AWS Glue 4.0. O Spark 3.3.0 tem um requisito mínimo de Log4j2, conforme mencionado aqui (https://spark.apache.org/docs/latest/core-migration-guide.html#upgrading-from-core-32-to-33).
- Várias atualizações de dependência, destacadas no apêndice.
- O Scala também foi atualizado para o 2.12 a partir do 2.11, e o Scala 2.12 não é compatível com o Scala 2.11.
- O Python 3.10 também é a versão padrão usada para scripts de Python, pois o AWS Glue 0.9 utilizava apenas o Python 2.
  - O Python 2.7 não é compatível com o Spark 3.3.0. Qualquer trabalho que solicite Python 2 na configuração do trabalho falhará com uma IllegalArgumentException.
  - Um novo mecanismo de instalação de módulos adicionais do Python por meio do pip está disponível.
- O AWS Glue 4.0 não é executado no Apache YARN, portanto, as configurações do YARN não se aplicam.
- O AWS Glue 4.0 não tem um Sistema de Arquivos Distribuído do Hadoop (HDFS).
- Quaisquer arquivos JAR extras fornecidos em trabalhos do AWS Glue 0.9 podem trazer dependências conflitantes, uma vez que houve atualizações em várias dependências no 3.0 a partir do 0.9. Você pode evitar conflitos de classpath no AWS Glue 3.0 com o parâmetro de trabalho `--user-jars-first` do AWS Glue.
- Agora o AWS Glue 4.0 é compatível com auto scaling. Portanto, a métrica ExecutorAllocationManager estará disponível quando o auto scaling estiver ativado.
- Em trabalhos do AWS Glue 4.0, você especifica o número de operadores e o tipo de operador, mas não especifica uma `maxCapacity`.
- O AWS Glue 4.0 ainda não é compatível com transformações de machine learning.
- Para migrar determinados conectores, consulte Migração de conectores e drivers JDBC para AWS Glue 4.0.
- O SDK de criptografia do AWS foi atualizado de 1.x para 2.x. Trabalhos do AWS Glue que usam configurações de segurança e trabalhos do AWS Glue dependentes da dependência do SDK de criptografia do AWS fornecida em runtime são afetados. Veja estas instruções para migração de trabalhos do AWS Glue.

- Você não pode migrar um trabalho de AWS Glue 0.9/1.0 para um trabalho do AWS Glue 4.0 diretamente. Isso ocorre porque, ao atualizar diretamente para a versão 2.x ou posterior e ativar todos os novos recursos imediatamente, o AWS Encryption SDK não conseguirá descriptografar o texto cifrado criptografado em versões anteriores do AWS Encryption SDK.
- Para atualizar com segurança, primeiro recomendamos que você migre para um ttrabalho do AWS Glue 2.0/3.0 que contenha a versão ponte do AWS Encryption SDK. Execute o trabalho uma vez para utilizar a versão ponte do AWS Encryption SDK.
- Após a conclusão, você pode migrar com segurança o trabalho do AWS Glue 2.0/3.0 para o AWS Glue 4.0.

Consulte a documentação de migração do Spark:

- Atualização do Spark SQL 2.2 para 2.3
- Atualização do Spark SQL 2.3 para 2.4
- Atualização do Spark SQL 2.4 para 3.0
- Atualização do Spark SQL 3.0 para 3.1
- Atualização do Spark SQL 3.1 para 3.2
- Atualização do Spark SQL 3.2 para 3.3
- Alterações no comportamento de data e hora a serem esperadas desde o Spark 3.0.

Migração de conectores e drivers JDBC para AWS Glue 4.0

Para as versões do JDBC e dos conectores de data lake que foram atualizadas, consulte:

- Apêndice B: upgrades do driver JDBC
- Apêndice C: Atualizações de conectores

Hudi

- Melhorias no suporte do Spark SQL:
  - Por meio do comando `Call Procedure`, há suporte adicional para atualização, downgrade, bootstrap, limpeza e reparo. A sintaxe de `Create/Drop/Show/Refresh Index` é possível no Spark SQL.

  - Foi eliminada uma discrepância de performance entre o uso por meio de um Spark DataSource em oposição ao Spark SQL. No passado, as gravações de fontes de dados costumavam ser mais rápidas do que em SQL.
  - Todos os geradores de chaves integrados implementam operações de API específicas do Spark com mais performance.
  - Substituiu a transformação UDF na operação de `insert` em massa com transformações RDD para reduzir os custos de uso do SerDe.
  - O Spark SQL com Hudi requer que seja especificada uma `primaryKey` por `tblproperites` ou opções na instrução SQL. Para operações de atualização e exclusão, o `preCombineField` também é necessário.
- Qualquer tabela Hudi criada antes da versão 0.10.0 sem uma `primaryKey` precisa ser recriada com um campo `primaryKey` desde a versão 0.10.0.

PostgreSQL

- Várias vulnerabilidades (CVEs) foram resolvidas.
- O Java 8 tem suporte nativo.
- Se o trabalho estiver usando matrizes de matrizes, com exceção das matrizes de bytes, esse cenário poderá ser tratado como matrizes multidimensionais.

MongoDB

- O conector atual do MongoDB é compatível com o Spark versão 3.1 ou posterior e o MongoDB versão 4.0 ou posterior.
- Devido à atualização do conector, alguns nomes de propriedades foram alterados. Por exemplo, o nome da propriedade de URI foi alterado para `connection.uri`. Para obter mais informações sobre as opções atuais, consulte o blog sobre o MongoDB Spark Connector.
- O uso do MongoDB 4.0 hospedado pelo Amazon DocumentDB tem algumas diferenças funcionais. Para obter mais informações, consulte estes tópicos:
  - Diferenças funcionais: Amazon DocumentDB e MongoDB
  - APIs, operações e tipos de dados compatíveis do MongoDB.
- A opção “particionador” é restrita a `ShardedPartitioner`, `PaginateIntoPartitionsPartitioner` e `SinglePartitionPartitioner`. Ela não pode usar `SamplePartitioner` e `PaginateBySizePartitioner` padrão para o Amazon

DocumentDB porque o operador de estágio não é compatível com a API do MongoDB. Para obter mais informações, consulte APIs, operações e tipos de dados compatíveis do MongoDB.

Delta Lake

- O Delta Lake agora é compatível com a viagem no tempo em SQL para consultar dados antigos com facilidade. Com essa atualização, a viagem no tempo agora está disponível no Spark SQL e por meio da API DataFrame. Foi adicionado suporte para a versão atual do TIMESTAMP em SQL.
- O Spark 3.3 apresenta o Trigger.AvailableNow para executar consultas de streaming como equivalente a `Trigger.Once` para consultas em lote. Esse suporte também está disponível ao usar tabelas Delta como fonte de streaming.
- Suporte a SHOW COLUMNS para retornar a lista de colunas em uma tabela.
- Suporte a DESCRIBE DETAIL na API Scala e Python DeltaTable. Ele recupera informações detalhadas sobre uma tabela Delta usando a API DeltaTable ou o Spark SQL.
- Suporte a retorno de métricas de operação dos comandos SQL Delete, Merge e Update. Anteriormente, esses comandos SQL retornavam um DataFrame vazio, agora eles retornam um DataFrame com métricas úteis sobre a operação executada.
- Melhorias de otimização de performance:
  - Defina a opção de configuração `spark.databricks.delta.optimize.repartition.enabled=true` para usar `repartition(1)` em vez de `coalesce(1)` no comando Optimize para melhorar a desempenho ao compactar muitos arquivos pequenos.
  - Desempenho aprimorado usando uma abordagem baseada em filas para paralelizar trabalhos de compactação.
- Outras alterações notáveis:
  - Suporte ao uso de variáveis nos comandos VACUUM e OPTIMIZE SQL.
  - Melhorias no CONVERT TO DELTA com tabelas de catálogo, incluindo:
    - Preenchimento automático do esquema de partição do catálogo quando ele não for fornecido.
    - Uso de informações de partição do catálogo para encontrar os arquivos de dados a serem confirmados em vez de fazer uma verificação completa do diretório. Em vez de confirmar todos os arquivos de dados no diretório da tabela, somente os arquivos de dados nos diretórios das partições ativas serão confirmados.

- Suporte a leituras em lote do Change Data Feed (CDF) em tabelas habilitadas para mapeamento de colunas quando DROP COLUMN e RENAME COLUMN não foram usados. Para obter mais informações, consulte a Documentação do CloudTrail Lake.
- Melhore a desempenho do comando Update ativando a remoção do esquema na primeira passada.

Apache Iceberg

- Foram adicionadas várias melhorias de performance para planejamento de varredura e consultas do Spark.
- Foi adicionado um cliente comum do catálogo REST que usa confirmações baseadas em alterações para resolver conflitos de confirmação no lado do serviço.
- A sintaxe `AS OF` para consultas SQL de viagem no tempo é compatível.
- Foi adicionado suporte de mesclagem na leitura para consultas MERGE e UPDATE.
- Foi adicionado suporte para reescrever partições usando a ordem Z.
- Foram adicionadas uma especificação e implementação para o Puffin, um formato para estatísticas grandes e blobs de índice, como esboços Theta ou filtros Bloom.
- Foram adicionadas novas interfaces para consumir dados de forma incremental (digitalizações de anexação e logs de alterações).
- Foi adicionado suporte para operações em massa e leituras variadas nas interfaces do FileIO.
- Foram adicionadas mais tabelas de metadados para mostrar arquivos excluídos na árvore de metadados.
- O comportamento de drop table mudou. No Iceberg 0.13.1, a execução de `DROP TABLE` remove a tabela do catálogo e também exclui o conteúdo da tabela. No Iceberg 1.0.0, `DROP TABLE` somente remove a tabela do catálogo. Para excluir o conteúdo da tabela, use `DROP TABLE PURGE`.
- As leituras vetorizadas de Parquet são habilitadas por padrão no Iceberg 1.0.0. Se você quiser desativar as leituras vetorizadas, defina `read.parquet.vectorization.enabled` como `false`.

Oracle

As mudanças são pequenas.

MySQL

As mudanças são pequenas.

Amazon Redshift

O AWS Glue 4.0 apresenta um novo conector do Amazon Redshift com um novo driver JDBC. Para obter informações sobre os aprimoramentos e como migrar de versões anteriores do AWS Glue, consulte the section called “Conexões do Redshift”.

Apêndice A: atualizações de dependência notáveis

Veja a seguir as atualizações de dependência:

Dependência	Versão no AWS Glue 4.0	Versão no AWS Glue 3.0	Versão no AWS Glue 2.0	Versão no 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 do AWS Glue Data Catalog	3.7.0	3.0.0	1.10.0	N/D
Python	3.10	3.7	2.7 e 3.6	2.7 e 3.6
Boto	1.26	1.18	1.12	N/D

Apêndice B: upgrades do driver JDBC

A seguir estão as atualizações do driver JDBC:

Driver	Versão do driver JDBC nas versões do AWS Glue anteriores	Versão do driver JDBC no AWS Glue 3.0	Versão do driver JDBC no 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
Bancos de dados da Oracle	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

Apêndice C: Atualizações de conectores

As atualizações do conector são as seguintes:

Driver	Versão do conector no AWS Glue 3.0	Versão do conector no AWS Glue 4.0
MongoDB	3.0.0	10.0.4
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

Migrar AWS Glue para Ray da versão prévia para o ambiente de runtime Ray2.4

> **Warning**
>
> Quando você salva seu trabalho do AWS Glue para Ray (versão prévia) no AWS Glue Studio, ele será automaticamente atualizado para o runtime do `Ray2.4`. Se você tiver problemas de compatibilidade com seu script, entre em contato com o suporte.

Você deve migrar trabalhos do AWS Glue que foram criados durante o AWS Glue para Ray (versão prévia) para o AWS Glue para Ray. Isso envolverá algumas mudanças simultâneas na configuração do seu trabalho.

- No campo `Runtime`, forneça o valor do runtime do `Ray2.4`. Isso atualizará a versão subjacente do Ray de 2.0.0 para 2.4.0.
- Certas bibliotecas Python incluídas por padrão na versão prévia não são mais fornecidas. Se seu trabalho aproveitar o SDK do AWS para pandas (awswrangler), dask, modin ou pymars, você precisará incluí-los como bibliotecas adicionais. Para obter mais informações sobre a inclusão de bibliotecas adicionais do Python, consulte. the section called “Módulos adicionais do Python para trabalhos do Ray”
- Se você estiver usando o parâmetro `--additional-python-modules`, os parâmetros usados para dar suporte a esse fluxo de trabalho foram divididos em `--pip-install` e `--s3-py-modules`. Para mais informações sobre esses parâmetros, consulte the section called “Módulos adicionais do Python para trabalhos do Ray”.
- Se você estiver usando o parâmetro `--auto-scaling-ray-min-workers`, ele foi renomeado como `--min-workers`.

Política de suporte a versão do AWS Glue

O AWS Glue é um serviço de integração de dados sem servidor que facilita a descoberta, preparação e combinação de dados para análise, machine learning e desenvolvimento de aplicações. Um trabalho do AWS Glue contém a lógica de negócios que executa o trabalho de integração de dados no AWS Glue. Há três tipos de trabalho no AWS Glue: Spark (lote e streaming), Ray e shell do Python. Ao definir o trabalho, você especifica a versão do AWS Glue que configura as

versões nos ambiente de runtime do Spark, Ray e Python subjacente. Por exemplo, um trabalho do Spark no AWS Glue versão 2.0 é compatível com Spark 2.4.3 e Python 3.7.

Política de suporte

Ocasionalmente o AWS Glue interrompe o suporte para antigas versões do AWS Glue. No entanto, os trabalhos em execução em versões obsoletas não são mais qualificados para receber suporte técnico. A AWS Glue não aplicará mais patches de segurança ou outras atualizações às versões obsoletas. A AWS Gluetambém não honrará SLAs quando os trabalhos forem executados em versões obsoletas.

Quando o suporte para o AWS Glue versão 2.0 ou posterior chegar ao fim, você não poderá mais criar trabalhos, mas apenas editar ou executar trabalhos.

As versões do AWS Glue a seguir atingiram ou estão programadas para término do suporte. O término do suporte começa à meia-noite (fuso horário do Pacífico) na data especificada.

Tipo	Versão do Glue	Término do suporte
Spark	Spark 2.2, Scala 2 (Glue versão 0.9)	1/6/2022
Spark	Spark 2.2, Python 2 (Glue versão 0.9)	1/6/2022
Spark	Spark 2.4, Python 2 (Glue versão 1.0)	1/6/2022
Spark	Spark 2.4, Python 3 (Glue versão 1.0)	30/9/2022
Spark	Spark 2.4, Scala 2 (Glue versão 1.0)	30/9/2022
Spark	Glue versão 2.0	31/1/2024
Tipo	Versão do Python	Término do suporte
Shell Python	Python 2 (Glue versão 1.0)	1/6/2022

Tipo	Versão do notebook	Término do suporte
Endpoint de desenvolvimento	Notebook do Zeppelin	30/9/2022

A AWS recomenda enfaticamente migrar seus trabalhos para versões compatíveis.

Para obter informações sobre como migrar trabalhos do Spark para a versão mais recente do AWS Glue, consulte Migrar trabalhos do AWS Glue para o AWS Glue versão 4.0.

Para migrar seus trabalhos de shell do Python para a versão mais recente do AWS Glue:

- No console, escolha `Python 3 (Glue Version 4.0)`.
- Na API CreateJob/UpdateJob, defina o parâmetro `GlueVersion` como `2.0` e `PythonVersion` como `3` no parâmetro `Command`. A configuração do `GlueVersion` não afeta o comportamento dos trabalhos de shell do Python, portanto, não há vantagem em incrementar o `GlueVersion`.
- Você precisa tornar seu script de trabalho compatível com o Python 3.

> **Note**
>
> Todas as regiões da AWS lançadas antes do lançamento da região de Jacarta, Indonésia (ap-southeast-3), em agosto de 2022, têm uma lista de permissão de clientes que podem realizar execuções de trabalhos no AWS Glue versão 0.9/1.0. Nessas regiões mais antigas, você pode criar um trabalho com um valor nulo e o padrão será a versão 0.9/1.0, dependendo da região. Para qualquer região da AWS lançada posteriormente, você deve definir explicitamente a versão do AWS Glue na API. O AWS Glue não aceita mais um parâmetro nulo. Se você passar 0.9 ou 1.0 no parâmetro, encontrará o erro "O Glue versão 0.9 (ou) 1.0 não e compatível".

Trabalhando com o Spark jobs em AWS Glue

Fornece informações sobre as AWS Glue tarefas de ETL do Spark.

Tópicos

- Parâmetros de trabalho do AWS Glue
- AWS Glue Spark e empregos PySpark

- Trabalhos de transmissão de ETL no AWS Glue
- Correspondência de registros com o FindMatches do AWS Lake Formation
- Migrar programas do Apache Spark para o AWS Glue

Parâmetros de trabalho do AWS Glue

Ao criar uma tarefa do AWS Glue, você define alguns campos padrão, como `Role` `WorkerType` e. Você pode fornecer informações adicionais de configuração por meio dos campos `Argument` (Job Parameters no console). Nesses campos, você pode fornecer aos trabalhos do AWS Glue os argumentos (parâmetros) listados neste tópico. Para obter mais informações sobre a API AWS Glue Job, consultethe section called “Tarefas”.

Definir parâmetros de trabalho

Você pode configurar um trabalho pelo console, na guia Job details (Detalhes do trabalho), no cabeçalho Job Parameters (Parâmetros do trabalho). Você também pode configurar um trabalho por meio da configuração AWS CLI by `DefaultArguments` ou `NonOverridableArguments` em um trabalho, ou `Arguments` configurando a execução de um trabalho. Os argumentos definidos no trabalho serão transmitidos toda vez que o trabalho for executado, enquanto os argumentos definidos na execução do trabalho serão transmitidos somente para essa execução individual.

Por exemplo, seguir temos a sintaxe para a execução de um trabalho usando `--arguments` para definir um parâmetro do trabalho.

```
$ aws glue start-job-run --job-name "CSV to CSV" --arguments='--scriptLocation="s3://
my_glue/libraries/test_lib.py"'
```

Acessar parâmetros do trabalho

Ao escrever scripts AWS Glue, talvez você queira acessar os valores dos parâmetros do trabalho para alterar o comportamento do seu próprio código. Fornecemos métodos auxiliares para fazer isso em nossas bibliotecas. Esses métodos resolvem os valores dos parâmetros de execução do trabalho que substituem os valores dos parâmetros do trabalho. Ao resolver parâmetros definidos em vários locais, o trabalho `NonOverridableArguments` substituirá a execução do trabalho `Arguments`, que substituirá o trabalho. `DefaultArguments`

No Python:

Em trabalhos do Python, fornecemos uma função denominada `getResolvedParameters`. Para ter mais informações, consulte the section called “getResolvedOptions”. Os parâmetros do trabalho estão disponíveis na variável `sys.argv`.

No Scala:

Em trabalhos do Scala, fornecemos um objeto denominado `GlueArgParser`. Para ter mais informações, consulte the section called “GlueArgParser”. Os parâmetros do trabalho estão disponíveis na variável `sysArgs`.

Referência de parâmetros de trabalho

O AWS Glue reconhece os seguintes nomes de argumento que podem ser usados para configurar o ambiente de script para os trabalhos e as execuções de trabalho:

`--additional-python-modules`

Uma lista delimitada por vírgulas representando um conjunto de pacotes do Python a serem instalados. Você pode instalar pacotes do PyPI ou fornecer uma distribuição personalizada. Uma entrada de pacote PyPI estará no formato *`package`*==*`version`*, com o nome PyPI e a versão do seu pacote de destino. Uma entrada de distribuição personalizada é o caminho do S3 para a distribuição.

As entradas usam a correspondência de versão do Python para combinar pacote e versão. Isso significa que você precisará usar dois sinais de igual, como ==. Existem outros operadores de correspondência de versão. Para obter mais informações, consulte PEP 440.

Para passar as opções de instalação do módulo para o `pip3`, use o parâmetro --python-modules-installer-option.

`--auto-scale-within-microbatch`

O valor padrão é falso. Esse parâmetro só pode ser usado para trabalhos de streaming do AWS Glue, que processam os dados de streaming em uma série de microlotes, e o escalonamento automático deve estar ativado. Ao definir esse valor como falso, ele calcula a média móvel exponencial da duração do lote para microlotes concluídos e compara esse valor com o tamanho da janela para determinar se o número de executores deve ser aumentado ou reduzido. O ajuste de escala só acontece quando um microlote é concluído. Ao definir esse valor como verdadeiro, durante um microlote, ele aumenta a escala verticalmente quando o número de tarefas do Spark permanece o mesmo por 30 segundos ou o processamento em lote atual é maior que o

tamanho da janela. O número de executores diminuirá se um executor ficar inativo por mais de 60 segundos ou se a média móvel exponencial da duração do lote for baixa.

`--class`

A classe Scala que serve como ponto de entrada para o seu script Scala. Isso se aplicará somente se `--job-language` for definido como `scala`.

`--continuous-log-conversionPattern`

Especifica um padrão personalizado de conversão de logs para um trabalho habilitado para registro em log contínuo. O padrão de conversão só se aplica a logs de driver e logs de executor. Isso não afeta a barra de progresso do AWS Glue.

`--continuous-log-logGroup`

Especifica um nome de grupo de CloudWatch log personalizado da Amazon para um trabalho habilitado para registro contínuo.

`--continuous-log-logStreamPrefix`

Especifica um prefixo de fluxo de CloudWatch log personalizado para um trabalho habilitado para registro contínuo.

`--customer-driver-env-vars` e **`--customer-executor-env-vars`**

Esses parâmetros definem variáveis de ambiente no sistema operacional, respectivamente, para cada operador (driver ou executor). Você pode usar esses parâmetros ao criar plataformas e estruturas personalizadas com base no AWS Glue, para permitir que seus usuários escrevam trabalhos em cima dele. A ativação desses dois sinalizadores permitirá definir variáveis de ambiente diferentes no driver e no executor, respectivamente, sem precisar injetar a mesma lógica no próprio script de trabalho.

Exemplo de uso

Este exemplo mostra como usar esses parâmetros:

```
"–customer-driver-env-vars", "CUSTOMER_KEY1=VAL1,CUSTOMER_KEY2=\"val2,val2 val2\"",
"–customer-executor-env-vars", "CUSTOMER_KEY3=VAL3,KEY4=VAL4"
```

Defini-los no argumento de execução do trabalho equivale a executar os seguintes comandos:

No driver:

- export CUSTOMER_KEY1=VAL1
- export CUSTOMER_KEY2="val2,val2 val2"

No executor:

- export CUSTOMER_KEY3=VAL3

Em seguida, no próprio script de trabalho, você pode recuperar as variáveis de ambiente usando `os.environ.get("CUSTOMER_KEY1")` ou `System.getenv("CUSTOMER_KEY1")`.

Sintaxe imposta

Observe os seguintes padrões ao definir variáveis de ambiente:

- Cada chave deve ter o `CUSTOMER_ prefix`.

  Por exemplo: para `"CUSTOMER_KEY3=VAL3,KEY4=VAL4"`, `KEY4=VAL4` será ignorado e não definido.
- Cada par de chave e valor deve ser delineado com uma única vírgula.

  Por exemplo: `"CUSTOMER_KEY3=VAL3,CUSTOMER_KEY4=VAL4"`
- Se o "valor" tiver espaços ou vírgulas, ele deverá ser definido entre aspas.

  Por exemplo: `CUSTOMER_KEY2=\"val2,val2 val2\"`

Essa sintaxe modela de perto os padrões de configuração de variáveis de ambiente bash.

`--datalake-formats`

Compatível com AWS Glue 3.0 e versões posteriores.

Especifica a estrutura do data lake a ser usada. AWS Glue adiciona os arquivos JAR necessários para as estruturas que você especifica no `classpath`. Para ter mais informações, consulte Usar estruturas de data lake com trabalhos do AWS Glue ETL.

Você pode especificar um ou mais dos seguintes valores, separados por uma vírgula:

- `hudi`
- `delta`
- `iceberg`

Por exemplo, passe o argumento a seguir para especificar todas as três estruturas.

```
'--datalake-formats': 'hudi,delta,iceberg'
```

`--disable-proxy-v2`

Desative o proxy de serviço para permitir chamadas AWS de serviço para o Amazon S3 e AWS Glue originadas de seu script por meio de sua VPC. CloudWatch Para mais informações, consulte Configurar chamadas do AWS para passarem pela VPC. Para desativar o proxy do serviço, defina o valor desse parâmetro como `true`.

`--enable-auto-scaling`

Ativa o recurso para usar escalação automática e faturamento por processador quando você define o valor como `true`.

`--enable-continuous-cloudwatch-log`

Habilita o registro em log contínuo em tempo real para trabalhos do AWS Glue. Você pode ver os registros de tarefas do Apache Spark em tempo real. CloudWatch

`--enable-continuous-log-filter`

Especifica um filtro padrão (`true`) ou nenhum filtro (`false`) quando você cria ou edita um trabalho habilitado para registro em log contínuo. Escolher o filtro padrão elimina o driver/executor desnecessário do Apache Spark e mensagens de log de heartbeat do Apache Hadoop YARN. Não escolher filtros fornece todas as mensagens de log.

`--enable-glue-datacatalog`

Permite que você use o AWS Glue Data Catalog como uma metastore do Apache Spark Hive. Para habilitar esse recurso, defina o valor como `true`.

`--enable-job-insights`

Permite o monitoramento adicional da análise de erros com o AWS Glue Job Run Insights. Para obter detalhes, consulte the section called “Monitoramento com insights de execuções de trabalho do AWS Glue”. Por padrão, o valor está definido como `true` e os insights da execução de trabalhos estão habilitados.

Essa opção só está disponível no AWS Glue versão 2.0 e 3.0.

`--enable-metrics`

Habilita a coleta de métricas para criar perfis de trabalhos para essa execução da tarefa. Essas métricas estão disponíveis no AWS Glue console e no CloudWatch console da Amazon. O valor

desse parâmetro não é relevante. Para ativar esse recurso, você pode fornecer esse parâmetro com qualquer valor, mas `true` é recomendado para maior clareza. Para desabilitar esse recurso, remova esse parâmetro da configuração do seu trabalho.

`--enable-observability-metrics`

Permite que um conjunto de métricas de observabilidade gere insights sobre o que está acontecendo dentro de cada trabalho executado na página Job Runs Monitoring no AWS Glue console e no Amazon CloudWatch console. Para habilitar esse recurso, defina o valor como true. Para desabilitar esse recurso, configure-o como `false` ou remova esse parâmetro da configuração do seu trabalho.

`--enable-rename-algorithm-v2`

Define a versão do algoritmo de renomeação do EMRFS como versão 2. Quando um trabalho do Spark usa o modo de substituição de partição dinâmica, existe a possibilidade de que uma partição duplicada seja criada. Por exemplo, você pode acabar com uma partição duplicada, como `s3://bucket/table/location/p1=1/p1=1`. Aqui, P1 é a partição que está sendo substituída. O algoritmo de renomeação versão 2 corrige esse problema.

Esta opção só está disponível no AWS Glue versão 1.0.

`--enable-s3-parquet-optimized-committer`

Habilita o committer otimizado para S3 do EMRFS a gravar dados Parquet no Amazon S3. É possível fornecer o par parâmetro/valor por meio do console do AWS Glue ao criar ou atualizar um trabalho do AWS Glue. Definir o valor como **`true`** habilitará o committer. Por padrão, a bandeira está ativada no AWS Glue 3.0 e desativada no AWS Glue 2.0.

Para obter mais informações, consulte Usar o committer otimizado para EMRFS S3.

`--enable-spark-ui`

Quando definido como `true`, ativa o recurso para usar a interface do usuário do Spark para monitorar e depurar trabalhos do AWS Glue ETL.

`--executor-cores`

Número de tarefas do Spark que podem ser executadas em paralelo. Essa opção é compatível com o AWS Glue 3.0+. O valor não deve exceder 2 vezes o número de vCPUs no tipo de operador, que é 8 no `G.1X`, 16 no `G.2X`, 32 no `G.4X` e 64 no `G.8X`. Você deve ter cuidado ao atualizar essa configuração, pois isso pode afetar a performance do trabalho, pois o aumento do paralelismo de tarefas gera pressão na memória e no disco, além aplicar o controle de utilização

nos sistemas de origem e de destino (por exemplo: causaria mais conexões simultâneas no Amazon RDS).

`--extra-files`

Os caminhos do Amazon S3 para arquivos adicionais, como arquivos de configuração, que o AWS Glue copia para o diretório de trabalho do seu script antes de executá-lo. Vários valores devem ser caminhos completos separados por uma vírgula (`,`). Somente arquivos individuais têm suporte, e não um caminho de diretório. Essa opção não é compatível com os tipos de trabalho do shell do Python.

`--extra-jars`

Os caminhos do Amazon S3 para arquivos Java `.jar` adicionais que o AWS Glue inclui no classpath Java antes de executar o script. Vários valores devem ser caminhos completos separados por uma vírgula (`,`).

`--extra-py-files`

Os caminhos do Amazon S3 para módulos adicionais do Python que o AWS Glue inclui no caminho do Python antes de executar seu script. Vários valores devem ser caminhos completos separados por uma vírgula (`,`). Somente arquivos individuais têm suporte, e não um caminho de diretório.

`--job-bookmark-option`

Controla o comportamento de um marcador de trabalho. Os seguintes valores de opção podem ser definidos.

Valor --job-bookmark-option	Descrição
`job-bookmark-enable`	Acompanhe os dados processados anteriormente. Quando um trabalho for executado, processe os novos dados desde o último ponto de verificação.
`job-bookmark-disable`	Sempre processe o conjunto de dados completo. Você é responsável por gerenciar o resultado das execuções de trabalho anteriores.
`job-bookmark-pause`	Processe dados incrementais desde a última execução bem-sucedida ou os dados no intervalo identificados pelas seguintes subopções, sem atualizar o estado do

Valor --job-bookmark-option	Descrição
	último marcador. Você é responsável por gerenciar o resultado das execuções de trabalho anteriores. As duas subopções são as seguintes: • **job-bookmark-from** `<from-value>` é o ID de execução que representa toda a entrada anterior que foi processada até a última execução bem-sucedida, incluindo o ID de execução especificado. A entrada correspondente é ignorada. • **job-bookmark-to** `<to-value>` é o ID de execução que representa toda a entrada anterior que foi processada até a última execução bem-sucedida, incluindo o ID de execução especificado. A entrada correspondente, exceto a entrada identificada pelo `<from-value>`, é processada pelo trabalho. Qualquer entrada depois dessa entrada também é excluída do processamento. O estado do marcador de trabalho não será atualizado quando esse conjunto de opções for especificado. As subopções são opcionais. No entanto, quando usadas, ambas as subopções devem ser fornecidas.

Para ativar um marcador de trabalho, por exemplo, transmita o seguinte argumento.

```
'--job-bookmark-option': 'job-bookmark-enable'
```

`--job-language`

A linguagem de programação de script. Esse valor deve ser `scala` ou `python`. Se esse parâmetro não estiver presente, o padrão será `python`.

`--python-modules-installer-option`

Uma string de texto não criptografado que define as opções a serem passadas para o `pip3` ao instalar módulos com --additional-python-modules. Forneça opções como você faria na linha de

comando, separadas por espaços e prefixadas por traços. Para obter mais informações sobre o uso, consulte the section called “Instalar módulos Python adicionais com pip no AWS Glue 2.0”.

> **Note**
>
> Essa opção não é compatível com trabalhos do AWS Glue quando você usa o Python 3.9.

`--scriptLocation`

O local do Amazon Simple Storage Service (Amazon S3) no qual seu script de ETL está localizado (no formato `s3://path/to/my/script.py`). Esse parâmetro substitui um local de script definido no objeto `JobCommand`.

`--spark-event-logs-path`

Especifica um caminho do Amazon S3. Ao usar o recurso de monitoramento da interface do usuário do Spark,o AWS Glue libera os logs de eventos do Spark para esse caminho do Amazon S3 a cada 30 segundos para um bucket que pode ser usado como um diretório temporário para armazenar eventos de interface do usuário do Spark.

`--TempDir`

Especifica um caminho do Amazon S3 para um bucket que pode ser usado como um diretório temporário para o trabalho.

Por exemplo, para definir um diretório temporário, transmita o seguinte argumento.

```
'--TempDir': 's3-path-to-directory'
```

> **Note**
>
> O AWS Glue criará um bucket temporário para trabalhos se um bucket ainda não existir em uma região. Esse bucket pode permitir o acesso público. Você pode modificar o bucket no Amazon S3 para definir o bloco de acesso público ou excluir o bucket mais tarde, depois que todos os trabalhos nessa região forem concluídos.

`--use-postgres-driver`

Ao definir esse valor como `true`, ele prioriza o driver JDBC do Postgres no classpath para evitar um conflito com o driver JDBC do Amazon Redshift. Essa opção só está disponível no AWS Glue versão 2.0.

`--user-jars-first`

Ao definir esse valor como `true`, ele prioriza os arquivos JAR extras do classpath. Essa opção só está disponível no AWS Glue versão 2.0 ou posterior.

`--conf`

Controla os parâmetros de configuração do Spark. Destina-se a casos de uso avançados.

`--encryption-type`

Parâmetro legado. O comportamento correspondente deve ser configurado usando configurações de segurança. Para obter mais informações sobre configurações de segurança, consulte the section called “Criptografar dados gravados pelo AWS Glue”.

O AWS Glue usa os seguintes argumentos internamente, e você nunca deve usá-los:

- `--debug`: interno do AWS Glue. Não o defina.
- `--mode`: interno do AWS Glue. Não o defina.
- `--JOB_NAME`: interno do AWS Glue. Não o defina.
- `--endpoint`: interno do AWS Glue. Não o defina.

O AWS Glue é compatível com a inicialização de um ambiente com o módulo `site` do Python usando `sitecustomize` para realizar personalizações específicas do site. A inicialização de suas próprias funções de inicialização só é recomendada para casos de uso avançados e é compatível na medida do possível com o AWS Glue 4.0.

O prefixo variável de ambiente, `GLUE_CUSTOMER`, é reservado para uso do cliente.

AWS Glue Spark e empregos PySpark

As seções a seguir fornecem informações sobre o AWS Glue Spark e as PySpark tarefas.

Tópicos

- Adicionar trabalhos Spark e PySpark no AWS Glue
- Rastrear dados processados usando marcadores de trabalho
- Gerenciador de ordem aleatória do Spark no AWS Glue com o Amazon S3
- Monitorar trabalhos Spark do AWS Glue

Adicionar trabalhos Spark e PySpark no AWS Glue

As seções a seguir fornecem mais informações sobre como adicionar trabalhos Spark e PySpark no AWS Glue.

Tópicos

- Configurando as propriedades da tarefa para tarefas do Spark no AWS Glue
- Editar scripts do Spark no console do AWS Glue
- Trabalhos (herdados)

Configurando as propriedades da tarefa para tarefas do Spark no AWS Glue

Um trabalho do AWS Glue encapsula um script que se conecta aos dados de origem, os processa e, depois, os grava no destino de dados. Normalmente, um trabalho executa scripts de extração, transformação e carga (ETL). Os trabalhos também podem executar scripts Python de uso geral (trabalhos do shell do Python). Os acionadores do AWS Glue podem iniciar trabalhos com base em uma programação ou um evento, ou sob demanda. É possível monitorar trabalhos para entender as métricas do runtime, status de conclusão, duração e hora de início.

Você pode usar scripts gerados pelo AWS Glue ou fornecer os seus próprios scripts. Com um esquema de origem e um local ou esquema de destino, o gerador de AWS Glue código pode criar automaticamente um script da API Apache Spark (). PySpark Você pode usar esse script como ponto de partida e editá-lo para atingir seus objetivos.

O AWS Glue pode escrever arquivos de saída em vários formatos de dados, incluindo JSON, CSV, ORC (Optimized Row Columnar), Apache Parquet e Apache Avro. Para alguns formatos de dados, é possível gravar formatos de compressão comuns.

Existem três tipos de trabalhos no AWS Glue: Spark, Streaming ETL e shell Python.

- Uma tarefa do Spark é executada em um ambiente Apache Spark gerenciado pelo. AWS Glue Ele processa os dados em lotes.

- Um trabalho de ETL de streaming é semelhante a um trabalho do Spark, exceto que ele executa ETL em streams de dados. Ele usa o framework do Apache Spark Structured Streaming. Alguns recursos de trabalho do Spark não estão disponíveis para trabalhos ETL de streaming.
- O trabalho do Shell do Python executa scripts Python como shell e é compatível com uma versão de Python que depende da versão do AWS Glue que você estiver usando. É possível usar esses trabalhos para programar e executar tarefas que não exigem um ambiente do Apache Spark.

Definir propriedades de trabalho para trabalhos do Spark

Ao definir o trabalho no console do AWS Glue, você fornece valores às propriedades para controlar o ambiente do runtime do AWS Glue.

A lista a seguir descreve as propriedades de um trabalho do Spark. Para as propriedades de um trabalho de shell do Python, consulte Definir propriedades de trabalho para trabalhos de shell Python. Para obter propriedades de um trabalho de ETL de streaming, consulte the section called “Definir propriedades de trabalho para um trabalho de ETL de transmissão”.

As propriedades estão listadas na ordem em que aparecem no assistente de Add job (Adicionar trabalho) no console do AWS Glue.

Nome

Forneça uma string UTF-8 com um tamanho máximo de 255 caracteres.

Descrição

Forneça uma descrição opcional da sua implantação com até 2048 caracteres.

Perfil do IAM

Especifique a função do IAM usada para a autorização de recursos necessários para a execução do trabalho e acesso aos armazenamentos de dados. Para obter mais informações sobre permissões de execução de trabalho no AWS Glue, consulte Gerenciamento de identidade e acesso para AWS Glue.

Tipo

O tipo de trabalho de ETL. Ele é definido automaticamente com base no tipo das fontes de dados selecionadas.

- Spark executa um script de ETL do Apache Spark com o comando de trabalho `glueetl`.

- Spark Streaming executa um script de ETL de streaming do Apache Spark com o comando de trabalho `gluestreaming`. Para ter mais informações, consulte the section called “Trabalhos de transmissão de ETL”.
- Shell do Python executa um script do Python com o comando de trabalho `pythonshell`. Para ter mais informações, consulte Configurar propriedades de trabalho para trabalhos de shell Python no AWS Glue.

versão do AWS Glue

A versão do AWS Glue determina as versões do Apache Spark e do Python que estão disponíveis para o trabalho, conforme especificado na tabela a seguir.

Versão do AWS Glue	Versões compatíveis do Spark e do Python
4,0	• Spark 3.3.0 • Python 3.10
3.0	• Spark 3.1.1 • Python 3.7
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 operador

Os seguintes tipos de operadores estão disponíveis:

Os recursos disponíveis para os AWS Glue trabalhadores são medidos em DPUs. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória.

- **`G.1X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador mapeia para 1 DPU (4 vCPUs, 16 GB de memória) com 84 GB de disco (aproximadamente 34 GB livres). Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
- **`G.2X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador mapeia para 2 DPU (8 vCPUs, 32 GB de memória) com 128 GB de disco (aproximadamente 77 GB livres). Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
- **`G.4X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador mapeia para 4 DPU (16 vCPUs, 64 GB de memória) com 256 GB de disco (aproximadamente 235 GB livres). Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark AWS nas seguintes regiões: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Cingapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central), Europa (Frankfurt), Europa (Irlanda) e Europa (Estocolmo).
- **`G.8X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador mapeia para 8 DPU (32 vCPUs, 128 GB de memória) com 512 GB de disco (aproximadamente 487 GB livres). Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark, nas mesmas AWS regiões compatíveis com o tipo de `G.4X` trabalhador.
- **`G.025X`**: ao escolher esse tipo, você também fornece um valor para Number of workers (Número de operadores). Cada operador mapeia para 0,25 DPU (2 vCPUs, 4 GB de memória) com 84 GB de disco (aproximadamente 34 GB livres). Recomendamos esse tipo de operador para trabalhos de streaming de baixo volume. Esse tipo de operador só está disponível para trabalhos de streaming AWS Glue versão 3.0.

É cobrada uma taxa por hora com base no número de DPUs usadas para executar os trabalhos de ETL. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

Com trabalhos do AWS Glue versão 1.0 ou anterior, quando você configura um trabalho usando o console e especifica um Worker type (Tipo de operador) como Standard (Padrão), a Maximum

capacity (Capacidade máxima) é definida e o Number of workers (Número de operadores) torna-se o valor de Maximum capacity (Capacidade máxima) - 1. Se você usar o AWS Command Line Interface (AWS CLI) ou o AWS SDK, poderá especificar o parâmetro de capacidade máxima ou especificar o tipo de trabalhador e o número de trabalhadores.

Para trabalhos do AWS Glue versão 2.0 ou posterior, não é possível especificar uma capacidade máxima. Em vez disso, você deve especificar um Worker type (Tipo de operador) e o Number of workers (Número de operadores).

Idioma

O código no script de ETL define a lógica do trabalho. O script pode ser codificado em Python ou Scala. Você pode escolher se o script que o trabalho executa é gerado pelo AWS Glue ou fornecido por você. É possível fornecer o nome e o local do script no Amazon Simple Storage Service (Amazon S3). Confirme se não existe um arquivo com o mesmo nome que o diretório do script no caminho. Para saber mais sobre como escrever scripts, consulte Guia de programação do AWS Glue.

O número solicitado de operadores

Para a maioria dos tipos de trabalho, é necessário especificar o número de operadores alocados quando o trabalho é executado.

Marcador de trabalho

Especifique como o AWS Glue processa essas informações de estado quando o trabalho é executado. É possível lembrar de dados previamente processados, atualizar informações de estado ou ignorar informações de estado. Para ter mais informações, consulte the section called “Rastrear dados processados usando marcadores de trabalho”.

Execução flexível

Ao configurar um trabalho usando o AWS Studio ou a API, você pode especificar uma classe de execução de trabalho padrão ou flexível. Seus trabalhos podem ter graus diversos de prioridade e sensibilidade ao tempo. A classe de execução padrão é ideal para workloads sensíveis ao tempo que exigem a inicialização rápida de trabalhos e recursos dedicados.

A classe de execução flexível é adequada para trabalhos não urgentes, como trabalhos de pré-produção, testes e cargas de dados únicas. Execuções flexíveis de trabalho são compatíveis com trabalhos que usem o AWS Glue versão 3.0 ou posterior e os tipos de operador `G.1X` ou `G.2X`.

As execuções flexíveis de trabalho são cobradas com base no número de operadores em execução a qualquer momento. É possível adicionar ou remover o número de operadores para

uma execução flexível de trabalho que esteja em andamento. Em vez de faturar como um simples cálculo de `Max Capacity` * `Execution Time`, cada operador contribuirá pelo tempo durante o qual foi executado na execução do trabalho. A conta é a soma de (`Number of DPUs per worker` * `time each worker ran`).

Para obter mais informações, consulte o painel de ajuda no AWS Studio ou Tarefas Execuções de trabalhos e.

Número de novas tentativas

Especifique o número de vezes, de 0 a 10, que o AWS Glue deve reiniciar automaticamente o trabalho em caso de falha. Os trabalhos que atingem o limite de tempo não são reiniciados.

Tempo limite de trabalho

Define o tempo máximo de execução em minutos. O padrão é 2.880 minutos (48 horas) para trabalhos em lotes. Quando o tempo de execução do trabalho excede esse limite, o estado da execução do trabalho é alterado para `TIMEOUT`.

Os trabalhos de streaming devem ter valores de tempo limite inferiores a 7 dias ou 100 a 80 minutos. Quando o valor for deixado em branco, o trabalho será reiniciado após 7 dias, caso você não tenha configurado uma janela de manutenção. Se você configurou uma janela de manutenção, ela será reiniciada durante a janela de manutenção após 7 dias.

> **Práticas recomendadas para tempos limite de trabalhos**
>
> Os trabalhos são cobrados com base no tempo de execução. Para evitar cobranças inesperadas, configure valores de tempo limite apropriados para o tempo de execução esperado do seu trabalho.

Propriedades avançadas

Nome do arquivo de script

Um nome de script exclusivo para seu trabalho. Não pode ser nomeado Trabalho sem título.

Caminho do script

A localização do Amazon S3 do script. O caminho deve estar no formato `s3://bucket/prefix/path/`. Ele deve terminar com uma barra (`/`) e não incluir arquivos.

Métricas de trabalho

Ative ou desative a criação de CloudWatch métricas da Amazon quando esse trabalho for executado. Para ver os dados de criação de perfil, você deve habilitar essa opção. Para obter mais informações sobre como ativar e visualizar as métricas, consulte Monitoramento e depuração de trabalho.

Métricas de observabilidade do trabalho

Ative a criação de CloudWatch métricas adicionais de observabilidade quando esse trabalho for executado. Para ter mais informações, consulte the section called “Monitoramento com métricas de observabilidade do AWS Glue”.

Registro em log contínuo

Ative o registro contínuo na Amazon CloudWatch. Se esta opção não estiver habilitada, os logs estarão disponíveis somente após o trabalho ser concluído. Para ter mais informações, consulte the section called “Registro contínuo para trabalhos do AWS Glue”.

IU do Spark

Ative o uso da interface do usuário do Spark para monitorar esse trabalho. Para ter mais informações, consulte Habilitar a interface do usuário da Web do Apache Spark para trabalhos do AWS Glue.

Caminho dos logs da interface do usuário do Spark

O caminho para gravar logs quando a interface do usuário do Spark está habilitada.

Configuração de log e monitoramento da interface do usuário do Spark

Escolha uma das seguintes opções:

- Padrão: grave registros usando o ID de execução do AWS Glue trabalho como nome do arquivo. Ative o monitoramento da interface do Spark no AWS Glue console.
- Legado: grave logs usando "spark-application- {timestamp}" como nome do arquivo. Não ative o monitoramento da interface do usuário do Spark.
- Padrão e legado: grave logs nos locais padrão e legados. Ative o monitoramento da interface do Spark no AWS Glue console.

Máximo de simultaneidade

Define o número máximo de execuções simultâneas permitidas para o trabalho. O padrão é um. Um erro será retornado quando este limite for atingido. O valor máximo que você pode

especificar é controlado por um limite de serviço. Por exemplo, se a execução anterior de um trabalho ainda estiver sendo realizada quando uma nova instância for iniciada, convém retornar um erro para evitar que duas instâncias do mesmo trabalho sejam executadas simultaneamente.

Caminho temporário

Informe o local de um diretório de trabalho no Amazon S3 onde os resultados intermediários temporários serão gravados quando o AWS Glue executar o script. Confirme se não existe um arquivo com o mesmo nome que o diretório temporário no caminho. Esse diretório é usado quando o AWS Glue lê e grava no Amazon Redshift e por determinadas transformações do AWS Glue.

> **Note**
>
> O AWS Glue criará um bucket temporário para trabalhos se um bucket ainda não existir em uma região. Esse bucket pode permitir o acesso público. Você pode modificar o bucket no Amazon S3 para definir o bloco de acesso público ou excluir o bucket mais tarde, depois que todos os trabalhos nessa região forem concluídos.

Limite de notificação de atraso (minutos)

Define o valor mínimo (em minutos) antes que uma notificação de atraso seja enviada. Você pode definir esse limite para enviar notificações quando uma execução de trabalho `RUNNING`, `STARTING` ou `STOPPING` levar mais do que o número de minutos esperado.

Configuração de segurança

Escolha uma configuração de segurança na lista. Uma configuração de segurança especifica como os dados no destino do Amazon S3 são criptografados: sem criptografia, criptografia no lado do servidor com chaves gerenciadas pelo AWS KMS(SSE-KMS) ou chaves de criptografia gerenciadas pelo Amazon S3 (SSE-S3).

Criptografia do lado do servidor

Se você selecionar essa opção, quando o trabalho de ETL gravar no Amazon S3, os dados serão criptografados em repouso usando criptografia SSE-S3. Tanto o seu destino de dados do Amazon S3 quanto outros dados gravados em um diretório temporário do Amazon S3 serão criptografados. Essa opção é passada como um parâmetro de trabalho. Para obter mais

informações, consulte Proteção de dados usando criptografia no lado do servidor com chaves de criptografia gerenciadas pelo Amazon S3 (SSE-S3) no Manual do usuário do Amazon Simple Storage Service.

> **Important**
>
> Esta opção será ignorada se uma configuração de segurança for especificada.

Usar o catálogo de dados do Glue como metastore do Hive

Selecione para usar o AWS Glue Data Catalog como a metastore do Hive. A função do IAM usada para o trabalho deve ter a permissão `glue:CreateDatabase`. Um banco de dados chamado “default” (padrão) é criado no Data Catalog, caso não exista.

Conexões

Escolha uma configuração de VPC para acessar fontes de dados do Amazon S3 localizadas na sua nuvem privada virtual (VPC). É possível criar e gerenciar a conexão de rede no AWS Glue. Para ter mais informações, consulte Conectar a dados.

Bibliotecas

Caminho da biblioteca Python, caminho dos JARs dependentes e caminho de arquivos referenciados

Especifique essas opções se o script precisar. É possível definir os caminhos do Amazon S3 separados por vírgulas para essas opções ao definir o trabalho. Você pode substituir esses caminhos ao executar o trabalho. Para ter mais informações, consulte Fornecer seus próprios scripts personalizados.

Parâmetros de trabalho

Um conjunto de pares de valor-chave que são transmitidos como parâmetros nomeados para o script. Esses são valores padrão que são usados quando o script é executado, mas é possível substituí-los em trigger ou ao executar o trabalho. É necessário iniciar o nome da chave com `--`; por exemplo: `--myKey`. Você passa os parâmetros do trabalho como um mapa ao usar AWS Command Line Interface o.

Para ver exemplos, consulte os parâmetros Python em Transmitir e acessar parâmetros de Python no AWS Glue.

Tags

Marque o trabalho com uma Tag key (Chave de tag) e um Tag value (Valor de tag) opcional. Depois que as chaves de tags são criadas, elas são somente leitura. Use as tags em alguns recursos para ajudar a organizá-los e identificá-los. Para ter mais informações, consulte Etiquetas da AWS no AWS Glue.

Restrições para trabalhos que acessam tabelas gerenciadas pelo Lake Formation

Lembre-se das seguintes notas e restrições ao criar trabalhos que leem ou gravam em tabelas gerenciadas por AWS Lake Formation:

- Não há suporte para os seguintes recursos em trabalhos que acessem tabelas com filtros no nível da célula:
  - Marcadores de trabalhos e execução limitada
  - Aplicação de predicados
  - Predicados de partição de catálogo do lado do servidor
  - enableUpdateCatalog

Editar scripts do Spark no console do AWS Glue

Um script contém o código que extrai dados das fontes, os transforma e os carrega em destinos. AWS Glue executa um script quando inicia um trabalho.

Os scripts de ETL do AWS Glue podem ser codificados em Python ou Scala. Os scripts Python usam uma linguagem que é uma extensão do dialeto PySpark Python para trabalhos de extração, transformação e carregamento (ETL). O script contém construtores estendidos para lidar com transformações de ETL. Quando você gera automaticamente a lógica do código fonte para um trabalho, o script é criado. Você pode editar este script ou fornecer seu próprio script para processar seu trabalho de ETL.

Para obter informações sobre como definir e editar scripts usando no AWS Glue, consulte Guia de programação do AWS Glue.

Bibliotecas ou arquivos adicionais

Se seu script precisar de bibliotecas ou arquivos adicionais, você poderá especificá-los da seguinte forma:

Caminho da biblioteca Python

Caminhos do Amazon Simple Storage Service (Amazon S3) separados por vírgulas para bibliotecas Python exigidas pelo script.

> **Note**
>
> Somente bibliotecas Python puras podem ser usadas. Bibliotecas que contam com extensões C, como a biblioteca de análise de dados Python pandas, ainda não são compatíveis.

Caminho de arquivos JAR dependentes

Caminhos do Amazon S3 separados por vírgulas para arquivos JAR exigidos pelo script.

> **Note**
>
> Atualmente, apenas bibliotecas Java ou Scala (2.11) podem ser usadas.

Caminho de arquivos referenciados

Caminhos do Amazon S3 separados por vírgula para arquivos adicionais (por exemplo, arquivos de configuração) exigidos pelo script.

Trabalhos (herdados)

Um script contém o código que executa o trabalho de extração, transformação e carregamento (ETL). Você pode fornecer seu próprio script, ou o AWS Glue pode gerar um script com a sua orientação sua. Para obter informações sobre como criar seus próprios scripts, consulte Fornecer seus próprios scripts personalizados.

Você pode editar um script no console do AWS Glue. Ao editar um script, você pode adicionar fontes, destinos e transformações.

Para editar um script

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/. Em seguida, escolha a guia Jobs.

2. Selecione um trabalho na lista e escolha Action, Edit script para abrir o editor de scripts.

   Você também pode acessar o editor de scripts na página de detalhes do trabalho. Escolha a guia Script e, em seguida, Edit script (Editar script).

Editor de scripts

O editor de scripts do AWS Glue permite inserir, modificar e excluir fontes, destinos e transformações no seu script. O editor de scripts exibe o script e um diagrama para ajudar você a visualizar o fluxo de dados.

Para criar um diagrama para o script, escolha Generate diagram (Gerar diagrama). O AWS Glue usa linhas de anotação no script que começam com ## para renderizar o diagrama. Para representar corretamente seu script no diagrama, você precisa manter em sincronia os parâmetros nas anotações e os parâmetros no código Apache Spark.

O editor de scripts permite que você adicione modelos de código sempre que o cursor estiver posicionado no script. Na parte superior do editor, escolha as seguintes opções:

- Para adicionar uma tabela de origem ao script, escolha Source.
- Para adicionar uma tabela de destino ao script, escolha Target.
- Para adicionar um local de destino ao script, escolha Target location.
- Para adicionar uma transformação ao script, escolha Transform. Para obter informações sobre as funções que são chamadas no seu script, consulte Programe scripts AWS Glue ETL em PySpark.
- Para adicionar uma transformação Spigot ao script, escolha Spigot.

No código inserido, modifique o `parameters` nas anotações e no código Apache Spark. Por exemplo, se você adicionar uma transformação Spigot, verifique se `path` foi substituído na linha de anotação `@args` e na linha de código `output`.

A guia Logs mostra os logs associados ao seu trabalho à medida que ele é executado. As mil linhas mais recentes são exibidas.

A guia Schema (Esquema) mostra o esquema das fontes e dos destinos selecionados, quando disponíveis no Data Catalog.

Rastrear dados processados usando marcadores de trabalho

O AWS Glue rastreia os dados que já foram processados durante uma execução anterior de um trabalho de ETL persistindo as informações do estado do trabalho executado. Essa informação de estado persistente é chamada de marcador de trabalho. Os marcadores do trabalho ajudam o AWS Glue a manter as informações de estado e a impedir o reprocessamento de dados antigos. Com marcadores de trabalho, você pode processar novos dados ao executar novamente em um intervalo programado. Um marcador de trabalho é composto pelos estados de vários elementos de trabalho, como fontes, transformações e destinos. Por exemplo, seu trabalho de ETL pode ler novas partições em um arquivo do Amazon S3. O AWS Glue rastreia as partições que o trabalho processou com êxito para evitar processamento e dados duplicados no armazenamento de dados de destino do trabalho.

Os marcadores de trabalho são implantados para fontes de dados JDBC, a transformação Relationalize e algumas fontes do Amazon Simple Storage Service (Amazon S3). A tabela a seguir lista os formatos de fonte do Amazon S3 aos quais o AWS Glue oferece suporte para marcadores de trabalho.

Versão do AWS Glue	Formatos de fonte do Amazon S3
Versão 0.9	JSON, CSV, Apache Avro, XML
Versão 1.0 e posterior	JSON, CSV, Apache Avro, XML, Parquet, ORC

Para obter mais informações sobre as versões do AWS Glue, consulte Definir propriedades de trabalho para trabalhos do Spark.

O atributo de marcadores de trabalhos tem funcionalidades adicionais quando acessado por meio de scripts do AWS Glue. Ao navegar pelo script gerado, você pode ver contextos de transformação relacionados a esse atributo. Para ter mais informações, consulte the section called “Usar marcadores de trabalho”.

Tópicos

- Usar marcadores de trabalho no AWS Glue
- Detalhes operacionais do atributo de marcadores de trabalho

Usar marcadores de trabalho no AWS Glue

A opção de marcador de trabalho é passada como um parâmetro quando o trabalho é iniciado. A tabela a seguir descreve as opções para configuração de marcadores de trabalho no console do AWS Glue.

Marcador de trabalho	Descrição
Enable (Habilitar)	Faz com que o trabalho atualize o estado após uma execução para controlar os dados processados anteriormente. Se seu trabalho tiver uma origem com suporte para marcador de trabalho, ele irá controlar os dados processados e, quando um trabalho for executado, processará novos dados desde o último ponto de verificação.
Disable (Desabilitar)	Os marcadores de trabalho não são usados, e o trabalho sempre processa todo o conjunto de dados. Você é responsável por gerenciar o resultado das execuções de trabalho anteriores. Esse é o padrão.
Pause	Processe dados incrementais desde a última execução bem-sucedida ou os dados no intervalo identificados pelas seguintes subopções, sem atualizar o estado do último marcador. Você é responsável por gerenciar o resultado das execuções de trabalho anteriores. As duas subopções são: • job-bookmark-from <from-value> é o ID de execução que represent a toda a entrada anterior que foi processada até a última execução bem-sucedida, incluindo o ID de execução especificado. A entrada correspondente é ignorada. • job-bookmark-to <to-value> é o ID de execução que representa toda a entrada anterior que foi processada até a última execução bem-suced ida, incluindo o ID de execução especificado. A entrada correspon dente, exceto a entrada identificada pelo <from-value>, é processad a pelo trabalho. Qualquer entrada depois dessa entrada também é excluída do processamento. O estado do marcador de trabalho não será atualizado quando esse conjunto de opções for especificado.

Marcador de trabalho	Descrição
	As subopções são opcionais. No entanto, quando usadas, é necessário fornecer as duas subopções.

Para obter detalhes sobre os parâmetros passados para um trabalho na linha de comando, e especificamente para marcadores de trabalho, consulte Parâmetros de trabalho do AWS Glue.

Para fontes de entrada do Amazon S3, os marcadores de trabalho do AWS Glue conferem a hora da última modificação dos objetos para verificar quais objetos precisam ser reprocessados. Se os dados da fonte de entrada tiverem sido modificados desde a última execução do trabalho, os arquivos serão reprocessados quando você executar o trabalho novamente.

Para fontes JDBC, as seguintes regras se aplicam:

- Para cada tabela, o AWS Glue usa uma ou mais colunas como chaves de marcadores para determinar dados novos e processados. As chaves de marcadores combinam-se para formar uma única chave composta.
- Por padrão, o AWS Glue usará a chave primária como chave de marcadores, desde que ela esteja aumentando ou diminuindo sequencialmente (sem lacunas).
- Você pode especificar as colunas a serem usadas como chaves de marcadores no seu script do AWS Glue. Para obter mais informações sobre como usar marcadores de trabalhos em scripts do AWS Glue, consulte the section called “Usar marcadores de trabalho”.
- O AWS Glue não oferece suporte ao uso de colunas com nomes que diferenciam maiúsculas e minúsculas como chaves de marcadores de trabalho.

É possível retroceder seus marcadores de trabalho de ETL do Spark do AWS Glue para qualquer execução de trabalho anterior. É possível oferecer melhor suporte a cenários de preenchimento ao retroceder os marcadores de trabalho para qualquer execução de trabalho anterior, resultando no reprocessamento de dados subsequente de execução de trabalho somente a partir da execução do trabalho marcado.

Para reprocessar todos os dados usando o mesmo trabalho, redefina o marcador do trabalho. Para redefinir o estado do marcador de trabalho, use o console do AWS Glue, a ResetJobBookmark ação (Python: reset_job_bookmark) operação da API ou a AWS CLI. Por exemplo, digite o comando a seguir usando a AWS CLI:

```
aws glue reset-job-bookmark --job-name my-job-name
```

Quando você retrocede ou redefine um marcador, o AWS Glue não limpa os arquivos de destino porque pode haver vários destinos e eles não são rastreados com marcadores de trabalho. Somente os arquivos de fontes são rastreados com marcadores de trabalho. Você pode criar diferentes destinos de saída ao retroceder e reprocessar os arquivos de fontes para evitar dados duplicados em sua saída.

O AWS Glue monitora os marcadores de trabalho por trabalho. Se você excluir um trabalho, o marcador dele será excluído.

Em alguns casos, você pode ter habilitado marcadores de trabalho do AWS Glue, mas o trabalho de ETL está reprocessando dados que já foram processados em uma execução anterior. Para obter informações sobre como resolver as causas comuns desse erro, consulte Solução de problemas de erros no AWS Glue Spark.

Detalhes operacionais do atributo de marcadores de trabalho

Esta seção descreve mais detalhes operacionais sobre o uso de marcadores de trabalho.

Os marcadores de trabalho armazenam os estados de um trabalho. Cada instância do estado é codificada por um nome e um número de versão do trabalho. Quando um script invoca `job.init`, ele recupera seu estado e sempre obtém a versão mais recente. Em um estado, há vários elementos de estado, que são específicos a cada origem, transformação e instância coletora no script. Esses elementos de estado são identificados por um contexto de transformação que é anexado ao elemento correspondente (origem, transformação ou coletor) no script. Os elementos do estado são salvos automaticamente quando `job.commit` é invocado no script do usuário. O script obtém o nome do trabalho e a opção de controle dos marcadores do trabalho nos argumentos.

Os elementos do estado no marcador do trabalho são origem, transformação ou dados específicos ao coletor. Por exemplo, suponha que você queira ler dados incrementais em um local do Amazon S3 que está sendo gravado constantemente por um trabalho ou processo upstream. Nesse caso, o script deve determinar o que foi processado até o momento. A implantação do marcador de trabalho para a fonte do Amazon S3 salva informações para que, quando o trabalho for executado novamente, ele possa filtrar somente os novos objetos usando as informações salvas e recalcular o estado para a próxima execução do trabalho. Um time stamp é usado para filtrar os novos arquivos.

Além dos elementos de estado, os marcadores de trabalho têm um número de execuções, um número de tentativas e um número de versão. O número de execução rastreia a execução do trabalho, e o número de tentativas registra as tentativas de uma execução de trabalho. O número de execuções do trabalho é um número incrementado de forma monotônica para cada execução bem-sucedida. O número de tentativas rastreia as tentativas de cada execução e só é incrementado quando há uma execução após uma tentativa com falha. O número da versão aumenta de forma monotônica e rastreia as atualizações em um marcador de trabalho.

No banco de dados do serviço AWS Glue, os estados de marcador para todas as transformações são armazenados juntos como pares de chave-valor:

```
{
  "job_name" : ...,
  "run_id": ...,
  "run_number": ..,
  "attempt_number": ...
  "states": {
    "transformation_ctx1" : {
      bookmark_state1
    },
    "transformation_ctx2" : {
      bookmark_state2
    }
  }
}
```

Práticas recomendadas

Veja a seguir as práticas recomendadas para usar marcadores de trabalhos.

- Não altere a propriedade origem dos dados com o marcador habilitado. Por exemplo, há um datasource0 apontando para um caminho de entrada A do Amazon S3, e o trabalho está lendo uma fonte que está sendo executada há várias rodadas com o marcador habilitado. Se você alterar o caminho de entrada de datasource0 para o caminho B do Amazon S3 sem alterar `transformation_ctx`, o trabalho do AWS Glue usará o antigo estado de marcador armazenado. Isso resultará em arquivos ausentes ou ignorados no caminho de entrada B, já que o AWS Glue presumiria que esses arquivos foram processados em execuções anteriores.
- Use uma tabela de catálogo com marcadores para um melhor gerenciamento de partições. Os marcadores funcionam tanto para origens de dados do Data Catalog quanto para origens de dados das opções. No entanto, é difícil remover/adicionar novas partições com a abordagem de opções.

Usar uma tabela de catálogo com crawlers pode fornecer melhor automação para rastrear as partições recém-adicionadas e proporcionar a flexibilidade para selecionar partições específicas com um predicado de aplicação.

- Use o compilador de listas de arquivos para Amazon S3 do AWS Glue para conjuntos de dados grandes. Um marcador listará todos os arquivos em cada partição de entrada e fará o arquivamento. Logo, se houver muitos arquivos em uma só partição, o marcador poderá entrar no driver OOM. Use o compilador de listas de arquivos para Amazon S3 do AWS Glue a fim de evitar listar todos os arquivos na memória de uma só vez.

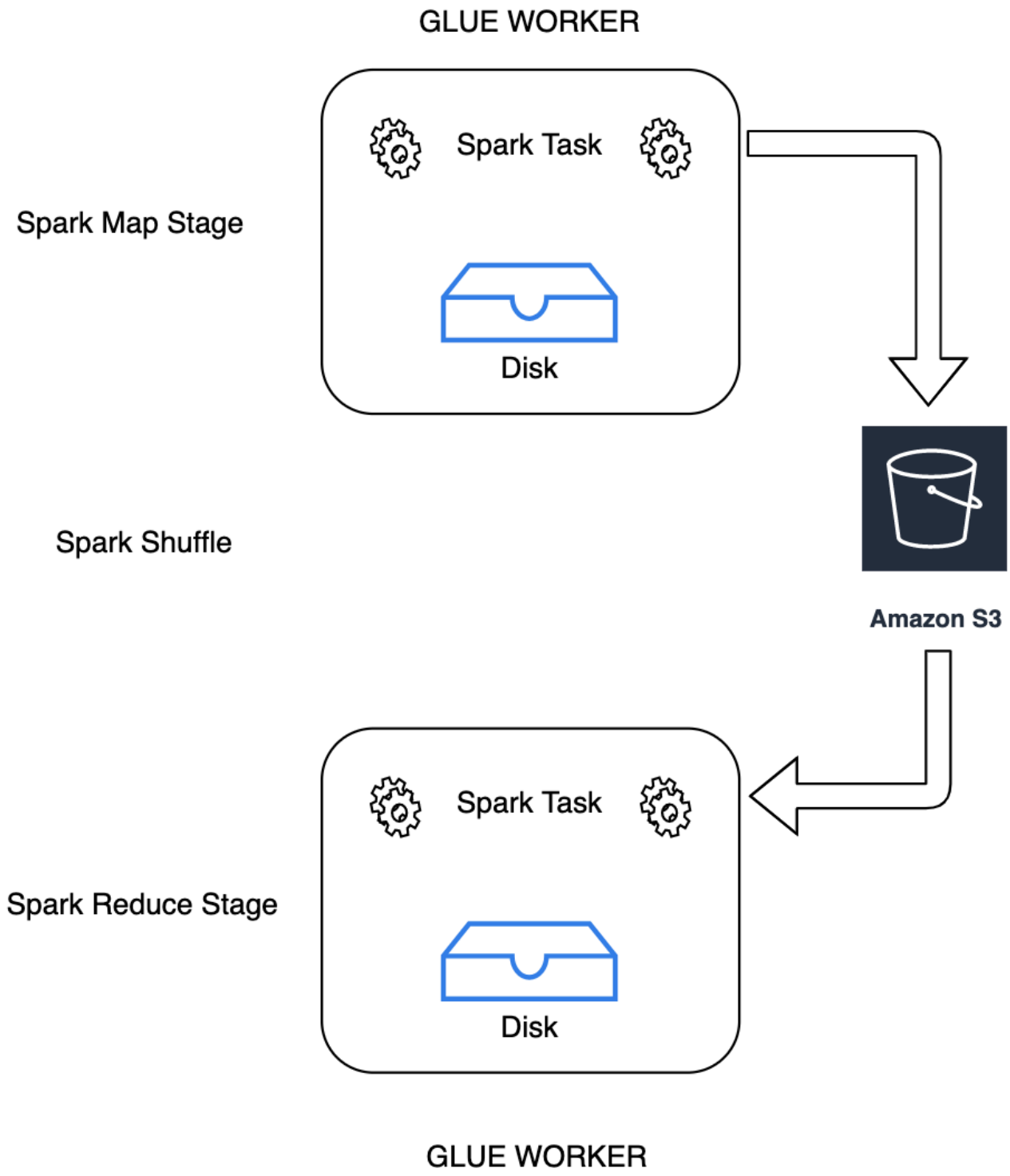
Gerenciador de ordem aleatória do Spark no AWS Glue com o Amazon S3

A colocação em ordem aleatória é uma etapa importante em um trabalho do Spark sempre que os dados são reorganizados entre partições. Isso é necessário porque transformações amplas, como `join`, `groupByKey`, `reduceByKey` e `repartition`, exigem informações de outras partições para concluir o processamento. O Spark reúne os dados necessários de cada partição e os combina em uma nova partição. Durante uma geração de ordem aleatória, os dados são gravados no disco e transferidos pela rede. Como resultado, a operação de ordem aleatória está vinculada à capacidade do disco local. O Spark lança os erros `No space left on device` ou `MetadataFetchFailedException` quando não há espaço em disco suficiente no executor e não há recuperação.

> **Note**
>
> O plugin AWS Glue Spark shuffle com Amazon S3 só é compatível com trabalhos de ETL do AWS Glue.

Solução

Com o AWS Glue, você pode usar o Amazon S3 para armazenar dados em ordem aleatória do Spark. O Amazon S3 é um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados, segurança e performance líderes do setor. Essa solução desagrega computação e armazenamento de seus trabalhos do Spark e oferece elasticidade completa e armazenamento em ordem aleatória de baixo custo, permitindo que você execute suas workloads mais intensas em ordem aleatória de forma confiável.



Estamos apresentando um novo Cloud Shuffle Storage Plugin for Apache Spark para usar o Amazon S3. Você pode ativar a ordem aleatória do Amazon S3 para executar trabalhos do AWS Glue de forma confiável e sem falhas, caso eles estejam vinculados à capacidade do disco local para grandes operações de ordem aleatória. Em alguns casos, a ordem aleatória para o Amazon S3 é ligeiramente

mais lenta do que o disco local (ou EBS), se você tiver um grande número de partições pequenas ou arquivos gravados em ordem aleatória no Amazon S3.

Pré-requisitos para usar o plug-in Cloud Shuffle Storage

Para usar o plug-in Cloud Shuffle Storage com trabalhos de ETL do AWS Glue, você precisa do seguinte:

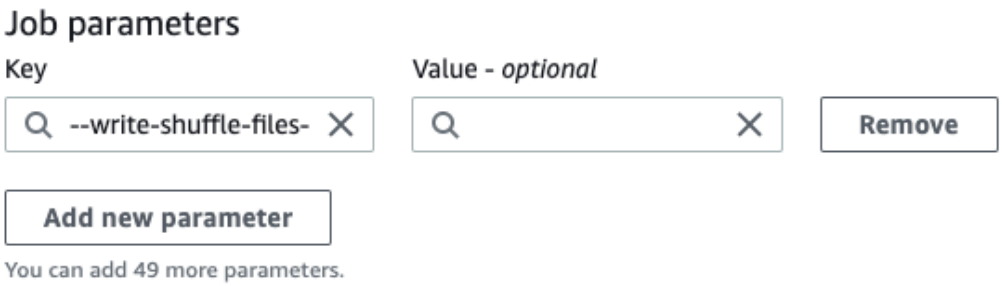
- Um bucket do Amazon S3 localizado na mesma região em que seu trabalho é executado, para armazenar dados intermediários embaralhados e vazados. O prefixo de armazenamento embaralhado do Amazon S3 pode ser especificado com o `--conf spark.shuffle.glue.s3ShuffleBucket=s3://`*`shuffle-bucket`*`/`*`prefix`*`/`, como no exemplo a seguir:

```
--conf spark.shuffle.glue.s3ShuffleBucket=s3://glue-shuffle-123456789-us-east-1/glue-shuffle-data/
```

- Defina as políticas de ciclo de vida de armazenamento do Amazon S3 no prefixo (como `glue-shuffle-data`), pois o gerenciador de ordem aleatória não limpa os arquivos após a conclusão do trabalho. O embaralhamento intermediário e os dados vazados devem ser excluídos após a conclusão do trabalho. Os usuários podem definir políticas de ciclo de vida curto no prefixo. Instruções para definir política do ciclo de vida do Amazon S3 estão disponíveis em Definir a configuração do ciclo de vida em um bucket no Guia do usuário do Amazon Simple Storage Service.

Usar o gerenciador de ordem aleatória do Spark no AWS Glue no Console da AWS

Para configurar o gerenciador de ordem aleatória do Spark no AWS Glue usando o console do AWS Glue ou o AWS Glue Studio ao configurar um trabalho: escolha o parâmetro de trabalho –write-shuffle-files-to-s3 para ativar a ordem aleatória do Amazon S3 para o trabalho.



Usar o plug-in de ordem aleatória do AWS Glue Spark

Os seguintes parâmetros de trabalho ativam e ajustam o gerenciador de ordem aleatória no AWS Glue. Esses parâmetros são sinalizadores, portanto, qualquer valor fornecido não será considerado.

- `--write-shuffle-files-to-s3`: o sinalizador principal que habilita o gerenciador de ordem aleatória do AWS Glue Spark para usar buckets do Amazon S3 a fim de gravar e ler dados em ordem aleatória. Quando o sinalizador não é especificado, o gerenciador de ordem aleatória não é usado.
- `--write-shuffle-spills-to-s3` - (compatível apenas com o AWS Glue versão 2.0). Um sinalizador opcional que permite que você descarregue arquivos de despejo em buckets do Amazon S3, o que fornece resiliência adicional ao seu trabalho do Spark. Isso só é necessário para grandes workloads que despejam muitos dados no disco. Quando o sinalizador não é especificado, nenhum arquivo de vazamento intermediário é gravado.
- `--conf spark.shuffle.glue.s3ShuffleBucket=s3://<shuffle-bucket>`: outro sinalizador opcional que especifica o bucket do Amazon S3 onde você grava os arquivos em ordem aleatória. Por padrão, `--TempDir`/shuffle-data. O AWS Glue 3.0+ suporta a gravação aleatória de arquivos em vários buckets especificando compartimentos com delimitador de vírgula, como em `--conf spark.shuffle.glue.s3ShuffleBucket=s3://`*`shuffle-bucket-1`*`/`*`prefix`*`,s3://`*`shuffle-bucket-2`*`/`*`prefix`*`/`. O uso de vários buckets melhora a performance.

É necessário fornecer configurações de segurança para habilitar a criptografia em repouso para os dados em ordem aleatória. Para obter mais informações sobre as configurações de segurança, consulte the section called “Configurar a criptografia”. O AWS Glue oferece suporte a todas as outras configurações relacionadas a ordem aleatória fornecidas pelo Spark.

Binários de software para o Cloud Shuffle Storage Plugin

Você também pode baixar os binários de software Cloud Shuffle Storage Plugin for Apache Spark sob a licença do Apache 2.0 e executá-lo em qualquer ambiente do Spark. O novo plug-in vem com suporte pronto para uso para o Amazon S3 e também pode ser facilmente configurado para usar outras formas de armazenamento em nuvem, como o Google Cloud Storage e o Microsoft Azure Blob Storage. Para obter mais informações, consulte Cloud Shuffle Storage Plugin for Apache Spark.

Observações e limitações

As seguintes são observações ou limitações sobre o gerenciador de ordem aleatória no AWS Glue:

- O AWS Glue shuffle manager não exclui automaticamente os arquivos de dados aleatórios (temporários) armazenados em seu bucket do Amazon S3 após a conclusão de um trabalho. Para garantir a proteção dos dados, siga as instruções no Pré-requisitos para usar o plug-in Cloud Shuffle Storage antes de ativar o plugin de Cloud Shuffle Storage.
- Você pode usar esse recurso se seus dados estiverem distorcidos.

Cloud Shuffle Storage Plugin for Apache Spark

O Cloud Shuffle Storage Plugin é um plug-in do Apache Spark compatível com a API `ShuffleDataIO` que permite armazenar dados em ordem aleatória nos sistemas de armazenamento em nuvem (como o Amazon S3). Ele ajuda você a complementar ou substituir a capacidade de armazenamento em disco local para grandes operações de ordem aleatória, como `join`, `reduceByKey`, `groupByKey` e `repartition`, nas aplicações do Spark, reduzindo assim as falhas comuns ou a depreciação de preço/performance dos dados de tecnologia sem servidor em trabalhos de análise e pipelines.

AWS Glue

As versões 3.0 e 4.0 do AWS Glue vêm com o plug-in pré-instalado e pronto para habilitar ordem aleatória para o Amazon S3 sem nenhuma etapa adicional. Para obter mais informações, consulte Gerenciador de ordem aleatória do Spark no AWS Glue com o Amazon S3 para ativar o recurso para as aplicações do Spark.

Outros ambientes do Spark

O plug-in exige que as seguintes configurações do Spark sejam definidas em outros ambientes do Spark:

- `--conf spark.shuffle.sort.io.plugin.class=com.amazonaws.spark.shuffle.io.cloud.Choppe` isso informa ao Spark que deve usar esse plug-in para E/S de ordem aleatória.
- `--conf spark.shuffle.storage.path=s3://`*`bucket-name`*`/`*`shuffle-file-dir`*: o caminho no qual os arquivos de ordem aleatória serão armazenados.

> **Note**
>
> O plug-in sobrescreve a classe principal do Spark. Como resultado, o jar do plugin precisa ser carregado antes dos jars do Spark. Você pode fazer isso usando `userClassPathFirst` em ambientes YARN locais se o plug-in for usado fora do AWS Glue.

Empacotar o plug-in com aplicações do Spark

Você pode empacotar o plug-in com aplicações do Spark e distribuições do Spark (versões 3.1 e superiores) adicionando a dependência do plug-in no `pom.xml` Maven enquanto desenvolve as aplicações do Spark localmente. Para obter mais informações sobre versões do Spark, consulte Versões do plug-in.

```
<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>
```

Ou então, você pode baixar os binários diretamente dos artefatos doAWS Glue Maven e incluí-los na aplicação do Spark como se segue.

```
#!/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/
```

Exemplo de 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> \
```

Configurações opcionais

Estas são as configurações opcionais que controlam o comportamento de ordem aleatória do Amazon S3.

- `spark.shuffle.storage.s3.enableServerSideEncryption`: habilite/desabilite o S3 SSE para arquivos de ordem aleatória e despejo. O valor padrão é `true`.
- `spark.shuffle.storage.s3.serverSideEncryption.algorithm`: o algoritmo de hash a ser usado. O valor padrão é `AES256`.
- `spark.shuffle.storage.s3.serverSideEncryption.kms.key`: o ARN da chave do KMS quando o SSE aws:kms está habilitado.

Junto com essas configurações, talvez seja necessário definir configurações como `spark.hadoop.fs.s3.enableServerSideEncryption` e outras configurações específicas do ambiente para garantir que a criptografia apropriada seja aplicada ao seu caso de uso.

Versões do plug-in

Esse plug-in é compatível com as versões do Spark associadas a cada versão do AWS Glue. A tabela a seguir mostra a versão do AWS Glue, a versão do Spark e a versão do plug-in associada, com o local do Amazon S3 para o binário de software do plug-in.

Versão do AWS Glue	Versão do Spark	Versão do plug-in	Local do 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/amazonaws/chopper-

Versão do AWS Glue	Versão do Spark	Versão do plug-in	Local do Amazon S3
			plugin/3.3-amzn-0/chopper-plugin-3.3-amzn-LATEST.jar

Licença

O binário do software para este plugin é licenciado sob a Licença do Apache-2.0.

Monitorar trabalhos Spark do AWS Glue

Tópicos

- Métricas do Spark disponíveis em AWS Glue Studio
- Monitorar trabalhos usando a interface do usuário da Web do Apache Spark
- Monitoramento com insights de execuções de trabalho do AWS Glue
- Monitorar com o Amazon CloudWatch
- Monitoramento e depuração de trabalho

Métricas do Spark disponíveis em AWS Glue Studio

A guia Metrics (Métricas) mostra métricas coletadas quando um trabalho é executado e a criação de perfil está ativada. Os seguintes gráficos são exibidos em trabalhos do Spark:

- Movimentação de dados ETL
- Perfil de memória: driver e executores

Escolha View additional metrics (Ver métricas adicionais) para mostrar os gráficos a seguir:

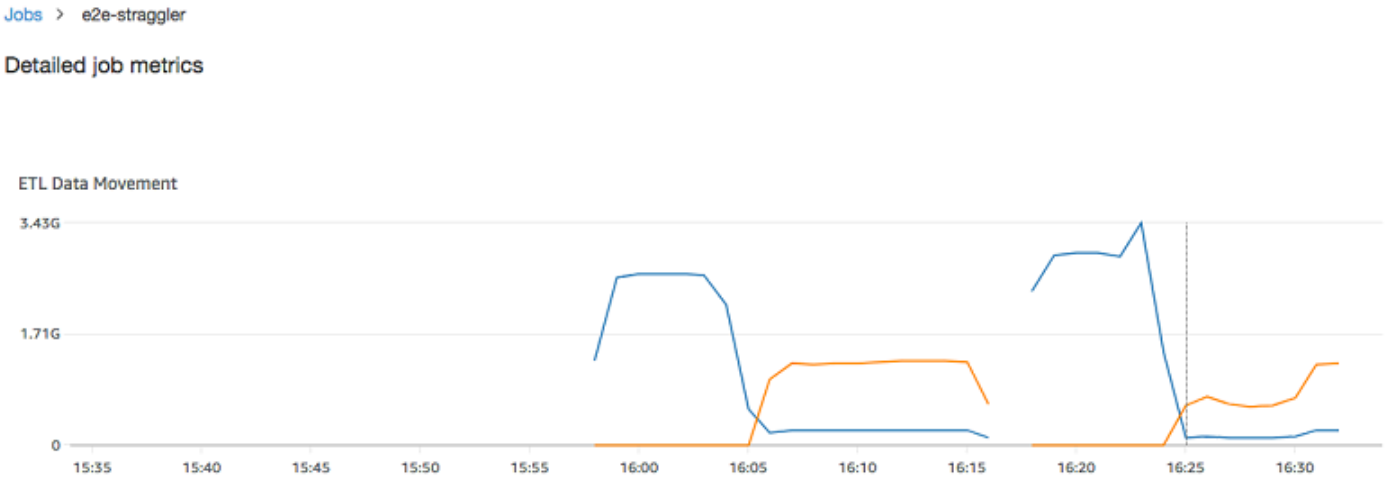
- Movimentação de dados ETL
- Perfil de memória: driver e executores
- Embaralhamento de dados em executores
- Carga da CPU: driver e executores
- Execução de trabalho: executores ativos, estágios concluídos e executores máximos necessários

Os dados desses gráficos são enviados para CloudWatch métricas se o trabalho estiver configurado para coletar métricas. Para obter mais informações sobre como ativar métricas e interpretar os gráficos, consulte Monitoramento e depuração de trabalho.

Example Grafo de movimentação de dados de ETL

O gráfico de movimentação de dados ETL mostra as seguintes métricas:

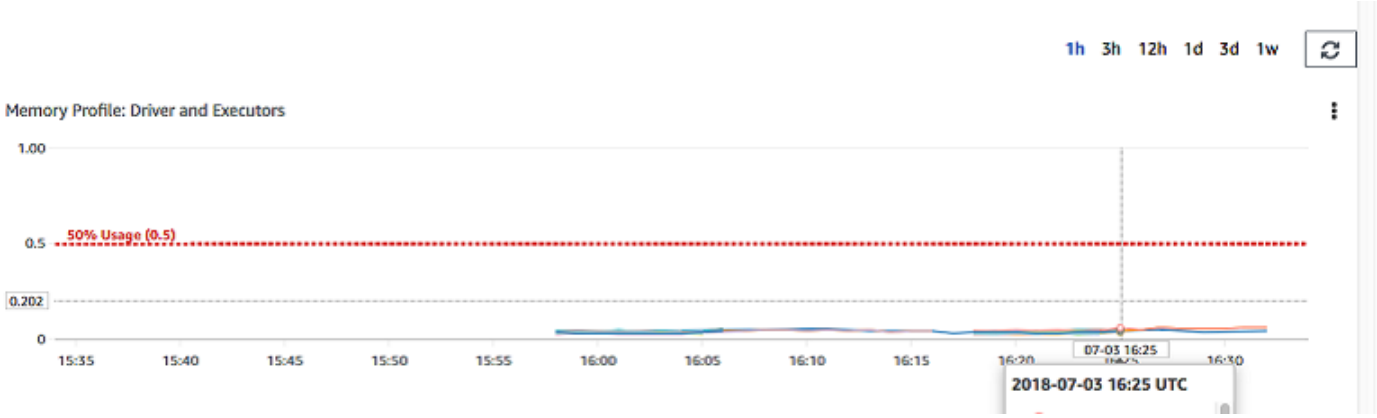
- O número de bytes lidos do Amazon S3 por todos os executores: `glue.ALL.s3.filesystem.read_bytes`
- O número de bytes gravados no Amazon S3 por todos os executores: `glue.ALL.s3.filesystem.write_bytes`



Example Grafo de perfil de memória

O gráfico de perfil de memória mostra as seguintes métricas:

- A fração de memória usada pelo heap da JVM para este driver (escala: 0–1) pelo driver, um executor identificado por executorId ou todos os executores:
  - `glue.driver.jvm.heap.usage`
  - `glue.executorId.jvm.heap.usage`
  - `glue.ALL.jvm.heap.usage`



Example Grafo de embaralhamento de dados em executores

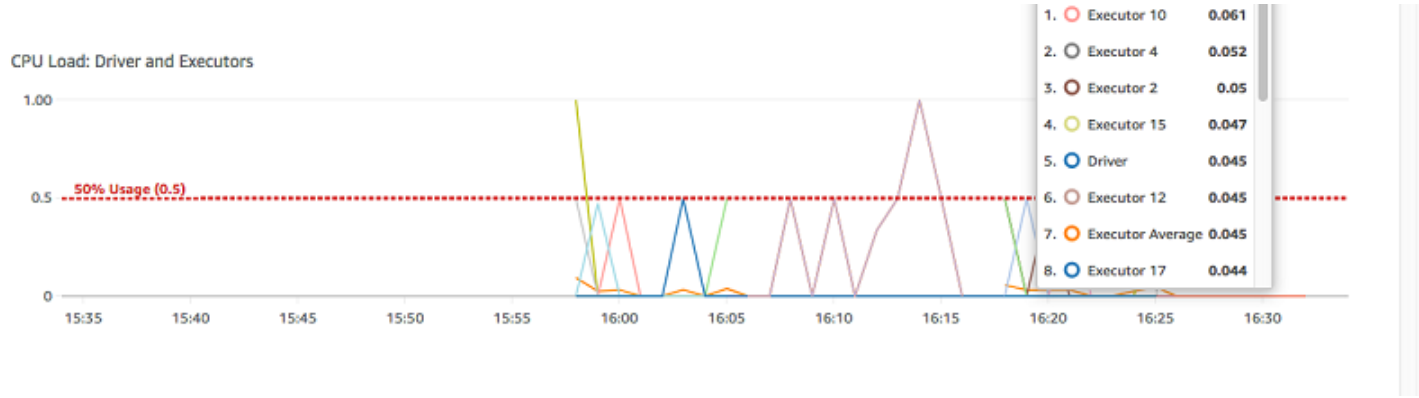
O gráfico de embaralhamento de dados nos executores mostra as seguintes métricas:

- O número de bytes lidos por todos os executores para embaralhar os dados entre eles — `glue.driver.aggregate.shuffleLocalBytesRead`
- O número de bytes gravados por todos os executores para embaralhar os dados entre eles — `glue.driver.aggregate.shuffleBytesWritten`

Example Grafo de carga de CPU

O gráfico de carga de CPU mostra as seguintes métricas:

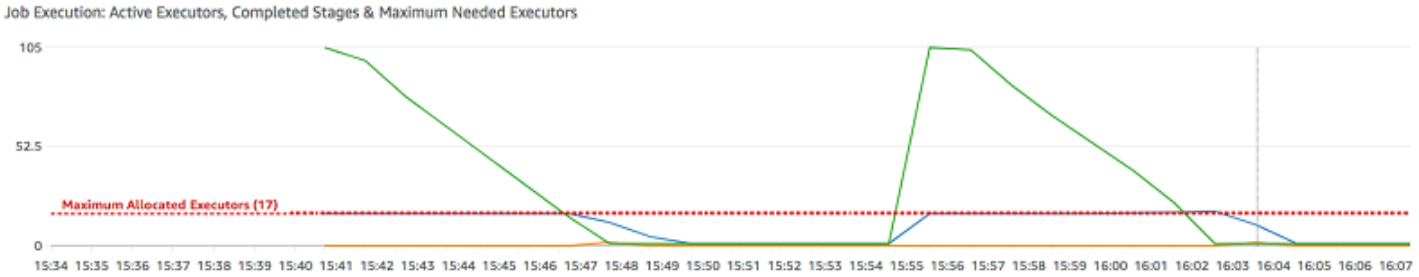
- A fração de carga de sistema da CPU usada (escala: 0–1) pelo driver, um executor identificado por executorId ou todos os executores:
  - `glue.driver.system.cpuSystemLoad`
  - `glue.`*`executorId`*`.system.cpuSystemLoad`
  - `glue.ALL.system.cpuSystemLoad`



Example Grafo de execução de trabalho

O gráfico de execução de trabalho mostra as seguintes métricas:

- O número de executores ativamente em execução — `glue.driver.ExecutorAllocationManager.executors.numberAllExecutors`
- O número de estágios concluídos — `glue.aggregate.numCompletedStages`
- O número máximo de executores necessários — `glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors`



Monitorar trabalhos usando a interface do usuário da Web do Apache Spark

Você pode usar a interface do usuário Web do Apache Spark para monitorar e depurar trabalhos de ETL do AWS Glue em execução no sistema de trabalhos do AWS Glue e também aplicativos Spark em execução em endpoints de desenvolvimento do AWS Glue. A interface do usuário do Spark permite que você verifique o seguinte para cada trabalho:

- O cronograma de eventos de cada estágio do Spark
- Um gráfico acíclico dirigido (DAG) do trabalho
- Planos físicos e lógicos para consultas SparkSQL

- As variáveis de ambiente do Spark subjacentes para cada trabalho

Para obter mais informações sobre como usar a interface do usuário Web do Spark, consulte Interface do usuário Web na documentação do Spark. Para obter orientação sobre como interpretar os resultados da interface do usuário do Spark para melhorar o desempenho do seu trabalho, consulte Melhores práticas para ajuste AWS Glue de desempenho para tarefas do Apache Spark em AWS Orientação prescritiva.

Você pode ver a interface do usuário do Spark no AWS Glue console. Isso está disponível quando um AWS Glue trabalho é executado em versões AWS Glue 3.0 ou posteriores com registros gerados no formato Padrão (em vez de legado), que é o padrão para trabalhos mais recentes. Se você tiver arquivos de log maiores que 0,5 GB, poderá ativar o suporte a registros contínuos para execuções de trabalhos em versões AWS Glue 4.0 ou posteriores para simplificar o arquivamento, a análise e a solução de problemas de registros.

Você pode ativar a interface do usuário do Spark usando o AWS Glue console ou o AWS Command Line Interface (AWS CLI). Quando você habilita a interface do usuário do Spark, os trabalhos de ETL do AWS Glue e as aplicações do Spark em endpoints de desenvolvimento do AWS Glue podem fazer backup dos logs de eventos do Spark em um local especificado por você no Amazon Simple Storage Service (Amazon S3). É possível usar os logs de eventos armazenados em backup no Amazon S3 com a interface do usuário do Spark em tempo real à medida que o trabalho é executado e após a conclusão do trabalho. Enquanto os registros permanecem no Amazon S3, a interface do usuário do Spark no AWS Glue console pode visualizá-los.

Permissões

Para usar a interface do Spark no AWS Glue console, você pode usar `UseGlueStudio` ou adicionar todas as APIs de serviço individuais. Todas as APIs são necessárias para aproveitar ao máximo a interface do usuário do Spark. No entanto, os usuários podem acessar os recursos da interface do usuário do Spark adicionando suas APIs de serviço à permissão do IAM para acesso refinado.

`RequestLogParsing` é o mais crítico, pois executa a análise dos logs. As APIs restantes destinam-se à leitura dos respectivos dados analisados. Por exemplo, `GetStages` fornece acesso aos dados sobre todas as etapas de um trabalho do Spark.

A lista de APIs de serviço da interface do usuário do Spark mapeadas em `UseGlueStudio` é mostrada no exemplo de política abaixo. A política abaixo fornece acesso para uso somente dos recursos da interface do usuário do Spark. Para adicionar mais permissões, como Amazon S3 e IAM, consulte Criação de políticas personalizadas do IAM para. AWS Glue Studio

A lista de APIs de serviço da interface do usuário do Spark mapeadas em `UseGlueStudio` é mostrada no exemplo de política abaixo. Ao usar uma API de serviço da interface do usuário do Spark, use o seguinte namespace: `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": [
    "*"
   ]
  }
 ]
}
```

Limitações

- A interface do usuário do Spark no AWS Glue console não está disponível para execuções de trabalhos que ocorreram antes de 20 de novembro de 2023 porque elas estão no formato de log antigo.
- A interface do usuário do Spark no AWS Glue console é compatível com registros contínuos para AWS Glue 4.0, como aqueles gerados por padrão em trabalhos de streaming. A soma máxima de todos os arquivos de eventos de log acumulados gerados é de 2 GB. Para AWS Glue trabalhos sem suporte a registros contínuos, o tamanho máximo do arquivo de eventos de log suportado pelo SparkUI é de 0,5 GB.
- A interface de usuário do Spark sem servidor não está disponível para registros de eventos do Spark armazenados em um bucket do Amazon S3 que só pode ser acessado pela sua VPC.

Exemplo: interface do usuário Web do Apache Spark

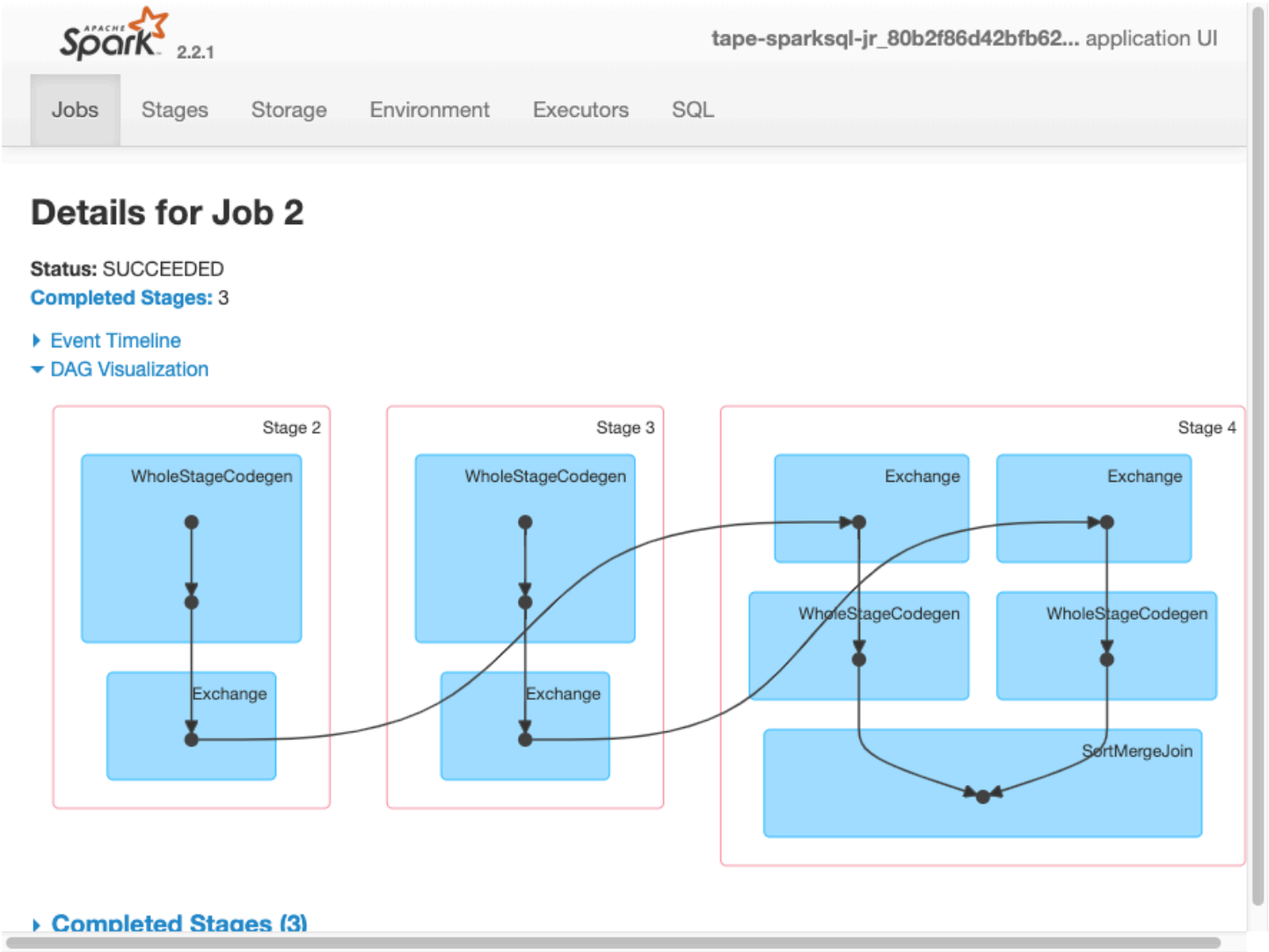
Este exemplo mostra como usar a interface do usuário do Spark para entender a performance do trabalho. As capturas de tela mostram a interface do usuário Web do Spark fornecida por um servidor de histórico autogerenciado do Spark. A interface do usuário do Spark no AWS Glue console oferece visualizações semelhantes. Para obter mais informações sobre como usar a interface do usuário Web do Spark, consulte Interface do usuário Web na documentação do Spark.

Veja a seguir um exemplo de uma aplicação Spark que lê de duas fontes de dados, realiza uma transformação de junção e a grava no Amazon S3 no formato Parquet.

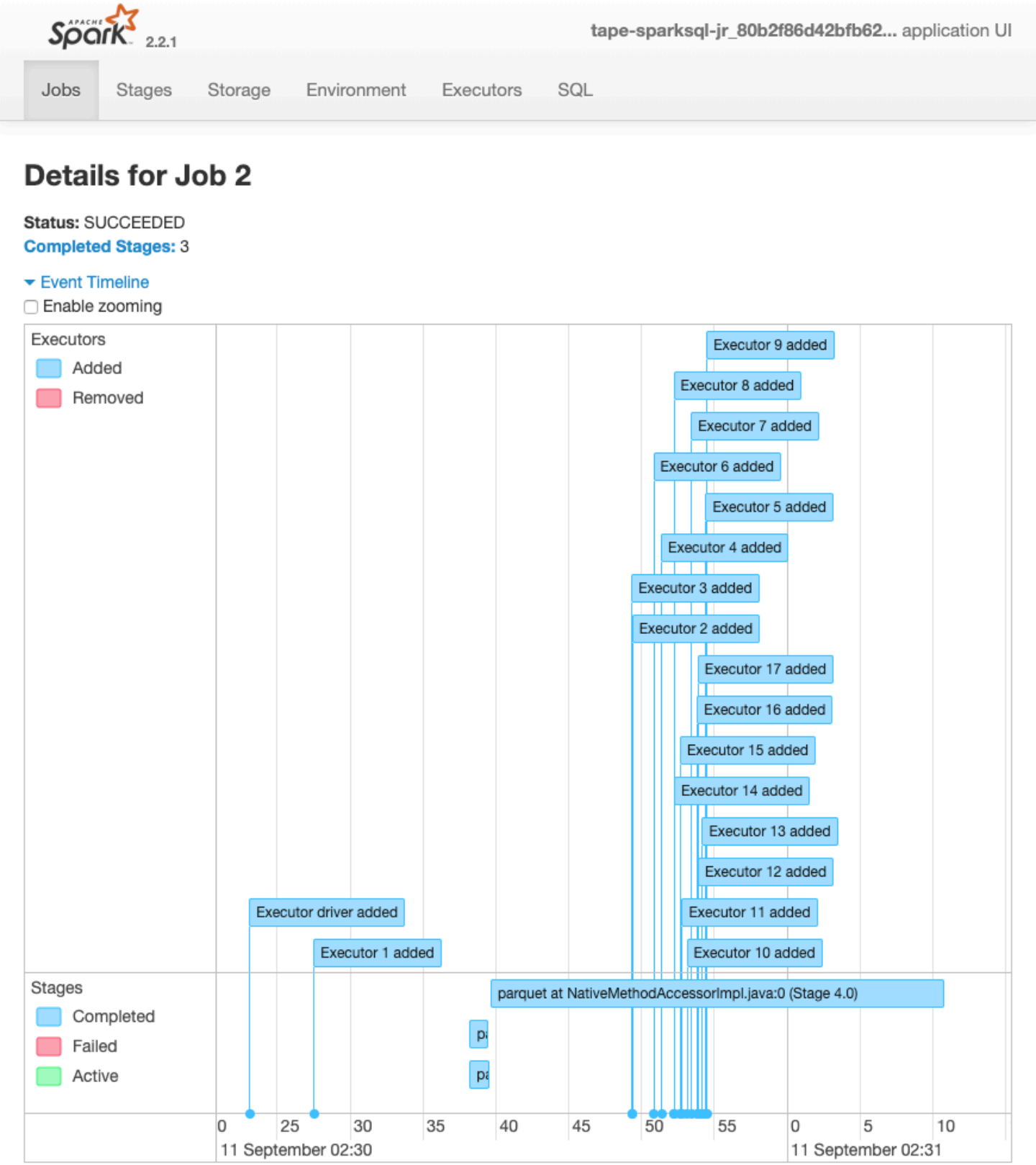
```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql.functions import count, when, expr, col, sum, isnull
from pyspark.sql.functions import 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()
```

A visualização do DAG a seguir mostra os diferentes estágios nesse trabalho do Spark.



O cronograma de eventos a seguir para um trabalho mostra o início, a execução e o encerramento de diferentes executores do 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)

A tela a seguir mostra os detalhes dos planos de consulta do SparkSQL:

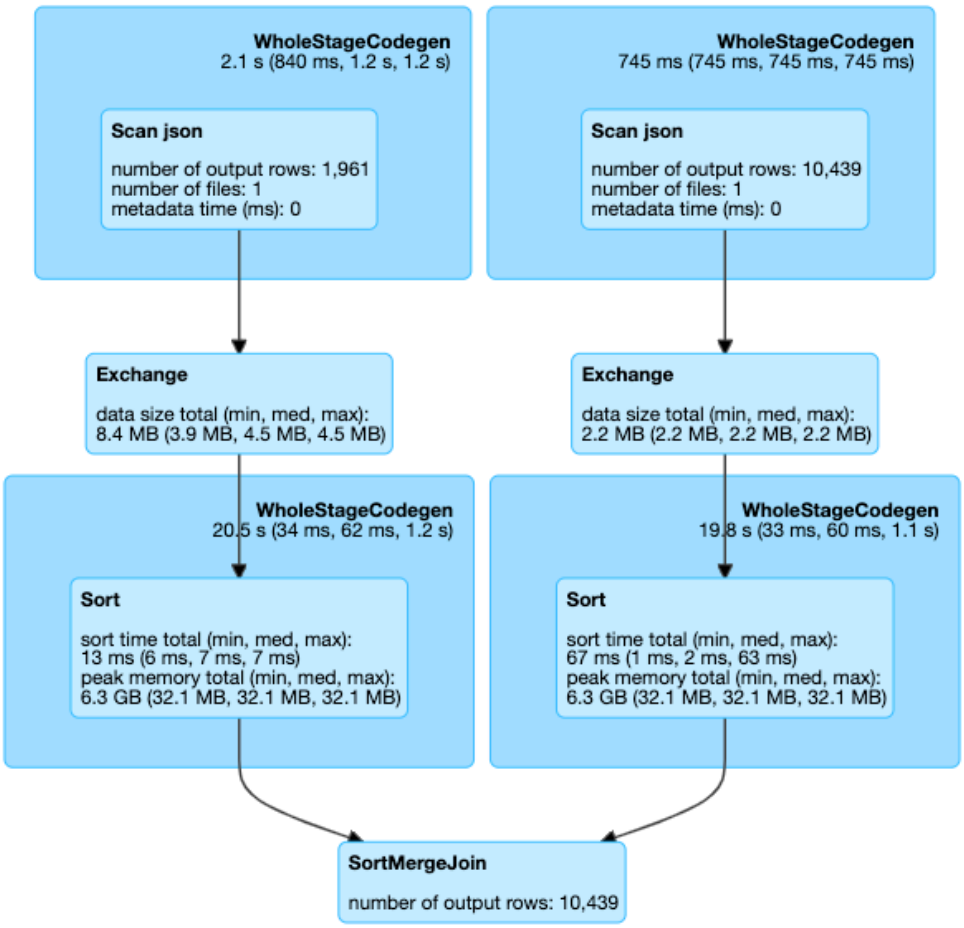
- Plano lógico examinado
- Plano lógico analisado
- Plano lógico otimizado
- Plano físico para execução

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)
```

Tópicos

Habilitar a interface do usuário da Web do Apache Spark para trabalhos do AWS Glue

Você pode usar a interface do usuário Web do Apache Spark para monitorar e depurar trabalhos de ETL do AWS Glue em execução no sistema de trabalhos do AWS Glue. Você pode configurar a interface do usuário do Spark usando o console do AWS Glue ou a AWS Command Line Interface (AWS CLI).

A cada 30 segundos, o AWS Glue faz backup dos logs de eventos do Spark para o caminho do Amazon S3 especificado.

Tópicos

Configurar a interface do usuário do Spark (console)

Siga estas etapas para configurar a interface do usuário do Spark usando o AWS Management Console. Ao criar um AWS Glue trabalho, a interface do usuário do Spark é ativada por padrão.

Para ativar a interface do Spark quando você cria ou edita um trabalho

1. Faça login no AWS Management Console e abra o AWS Glue console em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Tarefas.
3. Escolha Adicionar trabalho ou selecione um trabalho que já exista.
4. Em Detalhes do trabalho, abra as Propriedades avançadas.
5. Na guia Interface do usuário do Spark, escolha Gravar logs da interface do usuário do Spark no Amazon S3.
6. Especifique um caminho do Amazon S3 para armazenar os logs de eventos do Spark para o trabalho. Observe que, se você usar uma configuração de segurança no trabalho, a

criptografia também se aplicará ao arquivo de log da interface do usuário do Spark. Para ter mais informações, consulte Criptografar dados gravados pelo AWS Glue.

7. Em Configuração de log e monitoramento da interface do usuário do Spark:

   - Selecione Padrão se você estiver gerando registros para visualizar no AWS Glue console.
   - Selecione Legado se você estiver gerando logs para visualizar em um servidor de histórico do Spark.
   - Você também pode optar por gerar os dois.

Configurar a interface do usuário do Spark (AWS CLI)

Para gerar registros para visualização com a interface do usuário do Spark, no AWS Glue console, use o AWS CLI para passar os seguintes parâmetros de trabalho para AWS Glue trabalhos. Para ter mais informações, consulte the section called “Parâmetros de trabalho”.

```
'--enable-spark-ui': 'true',
'--spark-event-logs-path': 's3://s3-event-log-path'
```

Para distribuir logs para seus locais legados, defina o parâmetro `--enable-spark-ui-legacy-path` como `"true"`. Se não quiser gerar logs nos dois formatos, remova o parâmetro `--enable-spark-ui`.

Configurar a interface do usuário do Spark para sessões usando cadernos

> **Warning**
>
> AWS Glue No momento, as sessões interativas não oferecem suporte à interface do usuário do Spark no console. Configure um servidor de histórico do Spark.

Se você usa AWS Glue notebooks, configure o SparkUI antes de iniciar a sessão. Para fazer isso, use a célula da mágica `%%configure`:

```
%%configure { “--enable-spark-ui”: “true”, “--spark-event-logs-path”: “s3://path” }
```

Ativar registros contínuos

Habilitar o SparkUI e os arquivos de eventos de log contínuos para AWS Glue trabalhos oferece vários benefícios:

- Arquivos de eventos de registro contínuo — Com os arquivos de eventos de registro contínuo ativados, AWS Glue gera arquivos de log separados para cada etapa da execução do trabalho, facilitando a identificação e a solução de problemas específicos de um determinado estágio ou transformação.
- Melhor gerenciamento de registros — arquivos de eventos de log contínuos ajudam a gerenciar arquivos de log com mais eficiência. Em vez de ter um único arquivo de log potencialmente grande, os registros são divididos em arquivos menores e mais gerenciáveis com base nos estágios de execução do trabalho. Isso pode simplificar o arquivamento, a análise e a solução de problemas de registros.
- Maior tolerância a falhas — Se um AWS Glue trabalho falhar ou for interrompido, os arquivos de eventos de registro contínuo podem fornecer informações valiosas sobre o último estágio bem-sucedido, facilitando a retomada do trabalho a partir desse ponto, em vez de começar do zero.
- Otimização de custos — Ao ativar arquivos de eventos de log contínuos, você pode economizar nos custos de armazenamento associados aos arquivos de log. Em vez de armazenar um único arquivo de log potencialmente grande, você armazena arquivos de log menores e mais gerenciáveis, o que pode ser mais econômico, especialmente para trabalhos complexos ou de longa duração.

Em um novo ambiente, os usuários podem habilitar explicitamente os registros contínuos por meio de:

```
'–conf': 'spark.eventLog.rolling.enabled=true'
```

ou

```
'–conf': 'spark.eventLog.rolling.enabled=true –conf
spark.eventLog.rolling.maxFileSize=128m'
```

Quando os registros contínuos são ativados, `spark.eventLog.rolling.maxFileSize` especifica o tamanho máximo do arquivo de registro de eventos antes que ele seja transferido. O valor padrão desse parâmetro opcional, se não for especificado, é 128 MB. O mínimo é de 10 MB.

A soma máxima de todos os arquivos de eventos de log acumulados gerados é de 2 GB. Para AWS Glue trabalhos sem suporte a registros contínuos, o tamanho máximo do arquivo de eventos de log suportado pelo SparkUI é de 0,5 GB.

É possível desativar os logs contínuos de um trabalho de streaming por meio da passagem de uma configuração adicional. Observe que a manutenção de arquivos de log muito grandes pode ser cara.

Para desativar os logs contínuos, forneça a seguinte configuração:

```
'--spark-ui-event-logs-path': 'true',
'--conf': 'spark.eventLog.rolling.enabled=false'
```

Iniciar o servidor de histórico do Spark

É possível usar um servidor de histórico do Spark para visualizar os logs do Spark em sua própria infraestrutura. Você pode ver as mesmas visualizações no console do AWS Glue para execuções de trabalhos do AWS Glue no AWS Glue 4.0 ou versões posteriores com logs gerados no formato padrão (em vez de legado). Para ter mais informações, consulte the section called “Monitorar com a interface do usuário do Spark”.

Você pode iniciar o servidor de histórico do Spark usando um modelo do AWS CloudFormation que hospeda o servidor em uma instância do EC2 ou executar localmente usando o Docker.

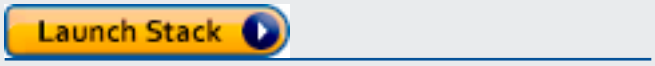
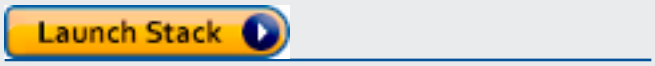
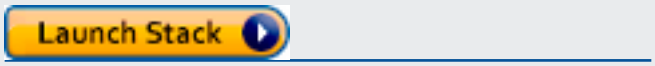
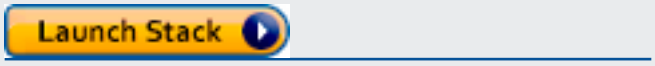
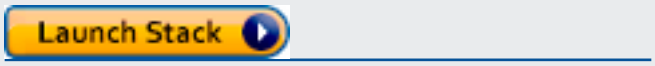
Tópicos

Iniciar o servidor de histórico do Spark e visualizar a interface do usuário do Spark usando o AWS CloudFormation

Você pode usar um modelo do AWS CloudFormation para iniciar o servidor de histórico do Apache Spark e visualizar a interface do usuário Web do Spark. Esses modelos são exemplos que você deve modificar para atender aos seus requisitos.

Como iniciar o servidor de histórico do Spark e visualizar a interface do usuário do Spark usando o AWS CloudFormation

1. Escolha um dos botões Launch Stack (Iniciar pilha) na tabela a seguir. Isso inicia a pilha no console do AWS CloudFormation.

Região	Executar
Leste dos EUA (Ohio)	
Leste dos EUA (N. da Virgínia)	
Oeste dos EUA (N. da Califórnia)	
Oeste dos EUA (Oregon)	
África (Cidade do Cabo)	
Ásia-Pacífico (Hong Kong)	
Ásia-Pacífico (Mumbai)	
Ásia-Pacífico (Osaka)	
Ásia-Pacífico (Seul)	
Ásia-Pacífico (Singapura)	
Ásia-Pacífico (Sydney)	
Ásia-Pacífico (Tóquio)	
Canadá (Central)	
Europa (Frankfurt)	

Região	Executar
Europa (Irlanda)	Launch Stack
Europa (Londres)	Launch Stack
Europa (Milão)	Launch Stack
Europa (Paris)	Launch Stack
Europa (Estocolmo)	Launch Stack
Oriente Médio (Bahrein)	Launch Stack
South America (São Paulo)	Launch Stack

2. Na página Specify template (Especificar modelo), escolha Next (Próximo).
3. Na página Specify stack details (Especificar detalhes da pilha), insira o Stack name (Nome da pilha). Insira informações adicionais em Parameters (Parâmetros).

   a. Configuração da interface do usuário do Spark

      Forneça as informações a seguir:

      - IP address range (Intervalo de endereços IP): o intervalo de endereços IP que pode ser usado para visualizar a interface do usuário do Spark. Se você deseja restringir o acesso de um intervalo de endereços IP específico, use um valor personalizado.
      - History server port (Porta do servidor de histórico): a porta da interface do usuário do Spark. Você pode usar o valor padrão.
      - Event log directory (Diretório de logs de evento): escolha o local onde os logs de eventos do Spark são armazenados nos endpoints de trabalho ou de desenvolvimento do AWS Glue. Você deve usar **`s3a://`** para o esquema de caminho dos logs de eventos.
      - Spark package location (Local do pacote do Spark): você pode usar o valor padrão.

- Keystore path (Caminho do repositório de chaves): o caminho do repositório de chaves SSL/TLS para HTTPS. Se você quiser usar um arquivo de armazenamento de chaves personalizado, especifique o caminho do S3 `s3://path_to_your_keystore_file` aqui. Se você deixar esse parâmetro vazio, um armazenamento de chaves baseado em certificado autoassinado será gerado e usado.
- Keystore password (Senha do repositório de chaves): insira uma senha do repositório de chaves SSL/TLS para HTTPS.

b. Configuração de instância do EC2

Forneça as informações a seguir:

- Instance type (Tipo de instância): o tipo de instância do Amazon EC2 que hospeda o servidor de histórico do Spark. Como esse modelo inicia a instância do Amazon EC2 em sua conta, o custo do Amazon EC2 será cobrado em sua conta separadamente.
- Latest AMI ID (ID da AMI mais recente): o ID da AMI do Amazon Linux 2 para a instância do servidor de histórico do Spark. Você pode usar o valor padrão.
- VPC ID (ID da VPC): o ID da nuvem privada virtual (VPC) da instância do servidor de histórico do Spark. Você pode usar qualquer uma das VPCs disponíveis em sua conta. Não é recomendado usar uma VPC padrão com uma Network ACL padrão. Para obter mais informações, consulte VPC padrão e sub-redes padrão e Criar uma VPC no Manual do usuário da Amazon VPC.
- Subnet ID (ID da sub-rede): o ID da instância do servidor de histórico do Spark. Você pode usar qualquer uma das sub-redes em sua VPC. Você deve ser capaz de acessar a rede do seu cliente para a sub-rede. Se quiser acessar pela Internet, você deverá usar uma sub-rede pública que tenha o gateway da Internet na tabela de rotas.

c. Escolha Próximo.

4. Na página Configure stack options (Configurar opções de pilha), para usar as credenciais do usuário atual para determinar como o CloudFormation pode criar, modificar ou excluir recursos na pilha, escolha Next (Próximo). Você também pode especificar um perfil na seção Permissões para usar em vez das permissões do usuário atual e, em seguida, escolher Próximo.
5. Na página Review (Revisar), revise o modelo.

   Selecione Reconheço que o AWS CloudFormation pode criar recursos do IAM e escolha Criar pilha.
6. Aguarde até que a pilha seja criada.

7. Abra a guia Outputs (Saídas).

   a. Copie o URL de SparkUiPublicUrl se você estiver usando uma sub-rede pública.

   b. Copie o URL de SparkUiPrivateUrl se você estiver usando uma sub-rede privada.

8. Abra um navegador da Web e cole o URL. Isso permite que você acesse o servidor usando HTTPS na porta especificada. É possível que seu navegador não reconheça o certificado do servidor. Se isso acontecer, substitua a proteção e prossiga.

Iniciar o servidor de histórico do Spark e visualizar a interface do usuário do Spark usando o Docker

Se você preferir acesso local (não ter uma instância do EC2 para o servidor de histórico do Apache Spark), também poderá usar o Docker para iniciar o servidor de histórico do Apache Spark e visualizar a interface do usuário do Spark localmente. Este Dockerfile é um exemplo que você deve modificar para atender aos seus requisitos.

Pré-requisitos

Para obter informações sobre como instalar o Docker em seu laptop, consulte a Comunidade do Docker Engine.

Como iniciar o servidor de histórico do Spark e visualizar a interface do usuário do Spark localmente usando o Docker

1. Faça download de arquivos do GitHub.

   Baixe o Dockerfile e o `pom.xml` dos exemplos de código do AWS Glue.

2. Determine se você deseja usar suas credenciais de usuário ou credenciais de usuário federado para acessar AWS.

   - Para usar as credenciais do usuário atual para acessar AWS, obtenha os valores para usar em `AWS_ACCESS_KEY_ID` e `AWS_SECRET_ACCESS_KEY` no comando `docker run`. Para obter mais informações, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do IAM.
   - Para usar usuários federados do SAML 2.0 para acessar AWS, obtenha os valores para `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, e `AWS_SESSION_TOKEN`. Para obter mais informações, consulte Solicitação de credenciais de segurança temporárias.

3. Determine a localização do diretório do log de eventos, a ser usado no comando `docker run`.

4. Crie a imagem do Docker usando os arquivos no diretório local, usando o nome `glue/sparkui` e a marcação `latest`.

```
$ docker build -t glue/sparkui:latest .
```

5. Crie e inicie o contêiner do Docker.

   Nos comandos a seguir, use os valores obtidos anteriormente nas etapas 2 e 3.

   a. Para criar o contêiner do Docker usando suas credenciais de usuário, use um comando semelhante ao seguinte

```
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 criar o contêiner do Docker usando credenciais temporárias, use `org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider` como provedor, e forneça os valores de credenciais obtidos na etapa 2. Para obter mais informações, consulte Uso de credenciais de sessão com TemporaryAWSCredentialsProvider na documentação Hadoop: Integração com a 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**
>
> Esses parâmetros de configuração vêm do Módulo Hadoop-AWS. Talvez seja necessário adicionar uma configuração específica com base em seu caso

de uso. Por exemplo: usuários em regiões isoladas precisarão configurar o `spark.hadoop.fs.s3a.endpoint`.

6. Abra `http://localhost:18080` no navegador para visualizar a interface do usuário do Spark localmente.

Monitoramento com insights de execuções de trabalho do AWS Glue

AWS Glueo job run insights é um recurso AWS Glue que simplifica a depuração e a otimização de trabalhos para seus trabalhos. AWS Glue AWS Gluefornece a interface do usuário do Spark, CloudWatch registros e métricas para monitorar seus AWS Glue trabalhos. Com esse recurso, você obtém as seguintes informações sobre a execução do seu AWS Glue trabalho:

- Número da linha do script do trabalho do AWS Glue que apresentou uma falha.
- Ação Spark executada por último no plano de consulta do Spark pouco antes da falha do seu trabalho.
- Eventos de exceção do Spark relacionados à falha apresentados em um fluxo de log ordenado ao longo do tempo.
- Análise de causa raiz e ação recomendada (como ajustar seu script) para corrigir o problema.
- Eventos comuns do Spark (mensagens de log relacionadas a uma ação do Spark) com uma ação recomendada que aborda a causa raiz.

Todos esses insights estão disponíveis para você usando dois novos fluxos de log nos CloudWatch registros de seus AWS Glue trabalhos.

Requisitos

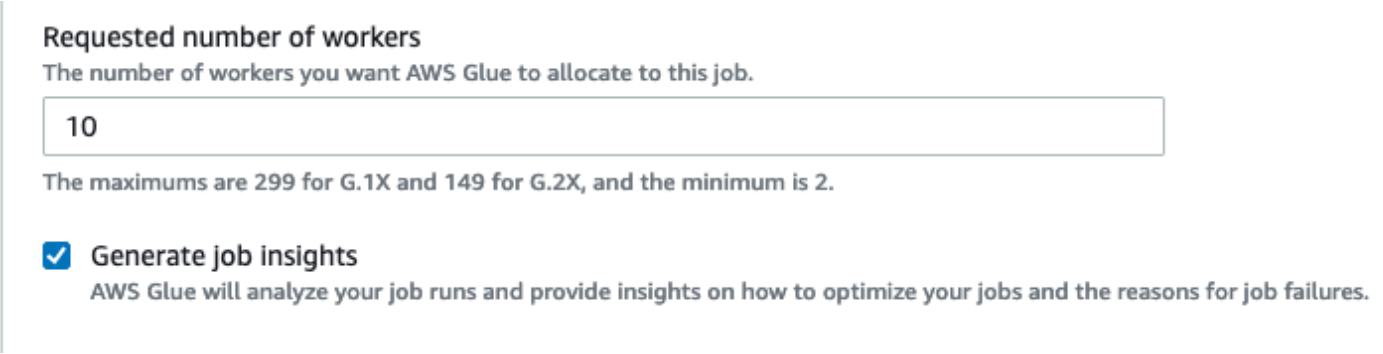
O recurso de insights de execução de AWS Glue tarefas está disponível para AWS Glue as versões 2.0, 3.0 e 4.0. Você pode seguir o guia de migração para seus trabalhos existentes para atualizá-los das versões mais antigas do AWS Glue.

Habilitando insights de execução de tarefas para uma tarefa AWS Glue de ETL

Você pode habilitar insights de execução de trabalhos por meio do AWS Glue Studio ou da CLI.

AWS Glue Studio

Ao criar um trabalho via AWS Glue Studio, você pode habilitar ou desabilitar insights de execuções de trabalhos na guia Job Details (Detalhes do trabalho). Verifique se a caixa Gerar informações sobre o trabalho está selecionada.



Linha de comando

Se estiver criando um trabalho por meio da CLI, você poderá iniciar uma execução de trabalho com um único novo parâmetro de trabalho: `--enable-job-insights = true`.

Por padrão, os fluxos de log de insights de execução de trabalhos são criados sob o mesmo grupo de logs padrão usado pelo log contínuo do AWS Glue, ou seja, `/aws-glue/jobs/logs-v2/`. Você pode configurar o nome do grupo de logs personalizado, filtros de log e configurações de grupo de logs usando o mesmo conjunto de argumentos para o log contínuo. Para obter mais informações, consulte Habilitar o log contínuo para trabalhos do AWS Glue.

Acessando o trabalho, execute os fluxos de registros do Insights em CloudWatch

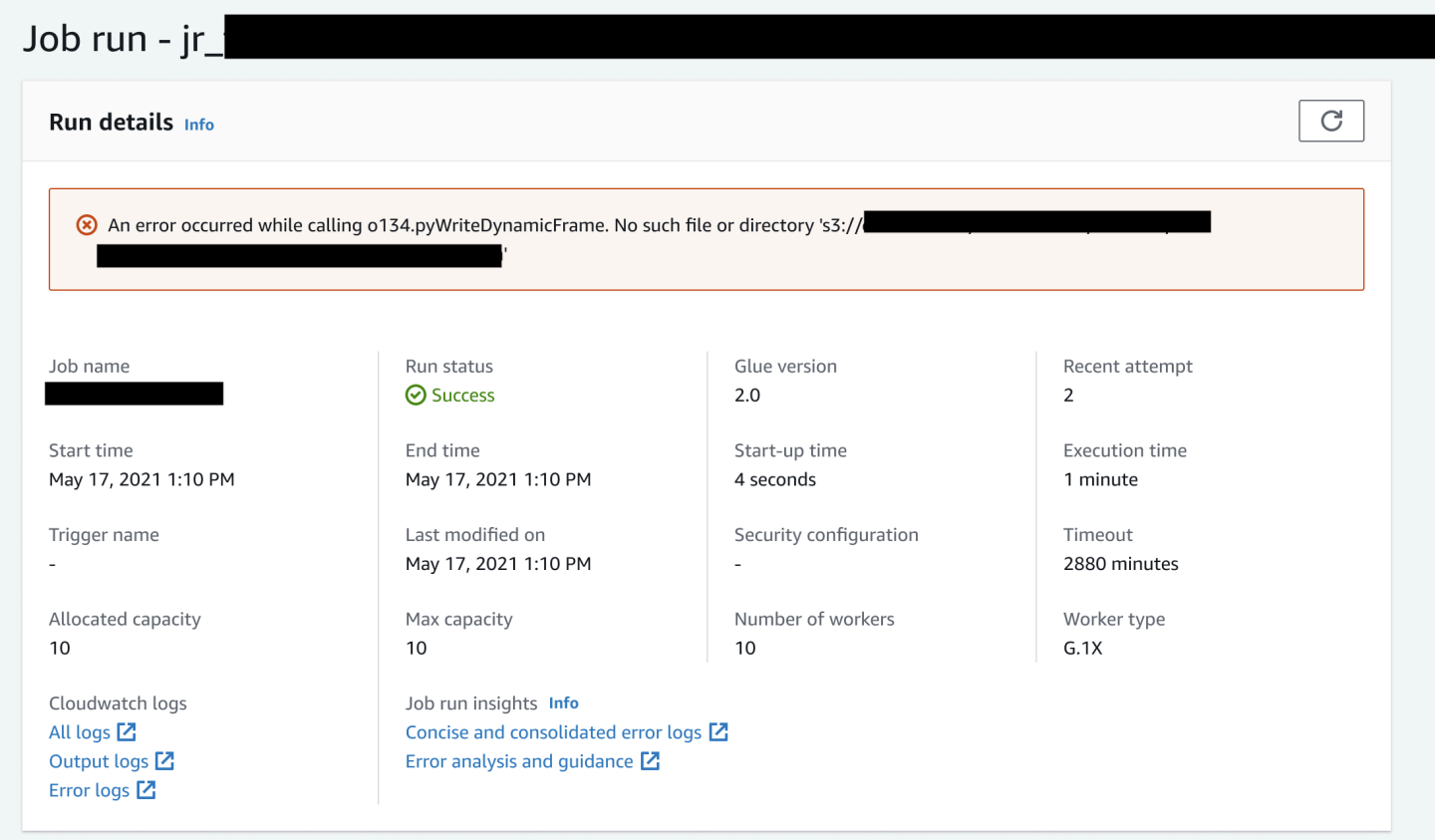
Com o recurso de insights de execução de trabalhos habilitado, pode haver dois fluxos de log criados quando uma execução de trabalho falha. Quando um trabalho termina com sucesso, nenhum dos fluxos é gerado.

1. Fluxo de log de análise de exceção: `<job-run-id>-job-insights-rca-driver`. Este stream fornece o seguinte:
   - Número da linha do script do trabalho do AWS Glue que causou a falha.
   - Ação do Spark executada por último no plano de consulta do Spark (DAG).
   - Eventos concisos ordenados por tempo do driver e executores do Spark relacionados à exceção. Você poderá encontrar detalhes como mensagens de erro completas, a tarefa do Spark com falha e os IDs dos executores para obter ajuda para se concentrar no fluxo de log do executor específico para uma investigação mais profunda, se necessário.

2. Fluxo de insights baseado em regras:
   - Análise de causa raiz e recomendações sobre como corrigir os erros (como usar um parâmetro de trabalho específico para otimizar a performance).
   - Eventos relevantes do Spark servindo de base para a análise de causa raiz e uma ação recomendada.

> **Note**
>
> O primeiro fluxo só existirá se qualquer evento de exceção do Spark estiver disponível para uma execução de trabalho com falha, e o segundo fluxo só existirá se houver insights disponíveis para a execução do trabalho com falha. Por exemplo, se o trabalho for concluído com êxito, nenhum dos fluxos será gerado; se o trabalho falhar, mas não houver uma regra definida por serviço que possa corresponder ao seu cenário de falha, somente o primeiro fluxo será gerado.

Se o trabalho for criado do AWS Glue Studio, os links para os fluxos acima também estarão disponíveis na guia de detalhes da execução do trabalho (insights da execução do trabalho) como "Concise and consolidated error logs" (Logs de erros concisos e consolidados) e "Error analysis and guidance" (Análise de erros e orientações).



Exemplo de insights de execuções de trabalhos do AWS Glue

Nesta seção, apresentamos um exemplo de como o recurso de insights de execução de trabalhos pode ajudar a resolver um problema em um trabalho que falhou. Neste exemplo, um usuário esqueceu de importar o módulo necessário (tensorflow) em um trabalho do AWS Glue para analisar e criar um modelo de machine learning em seus dados.

```
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
```

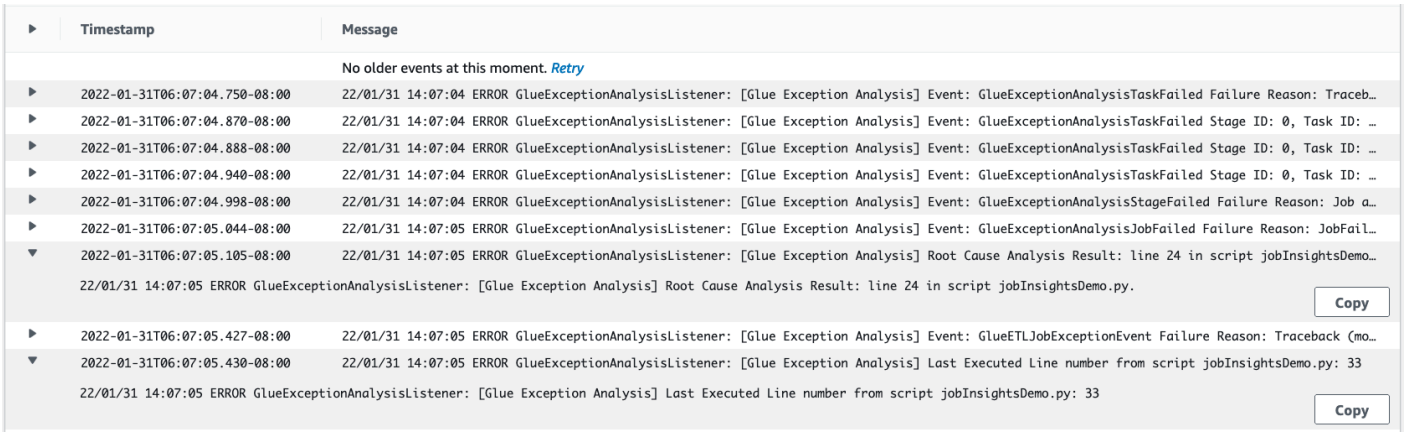
```
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

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())
```

Sem o recurso de insights de execução de trabalhos, quando o trabalho falhar, você só vê esta mensagem emitida pelo Spark:

`An error occurred while calling o111.collectToPython. Traceback (most recent call last):`

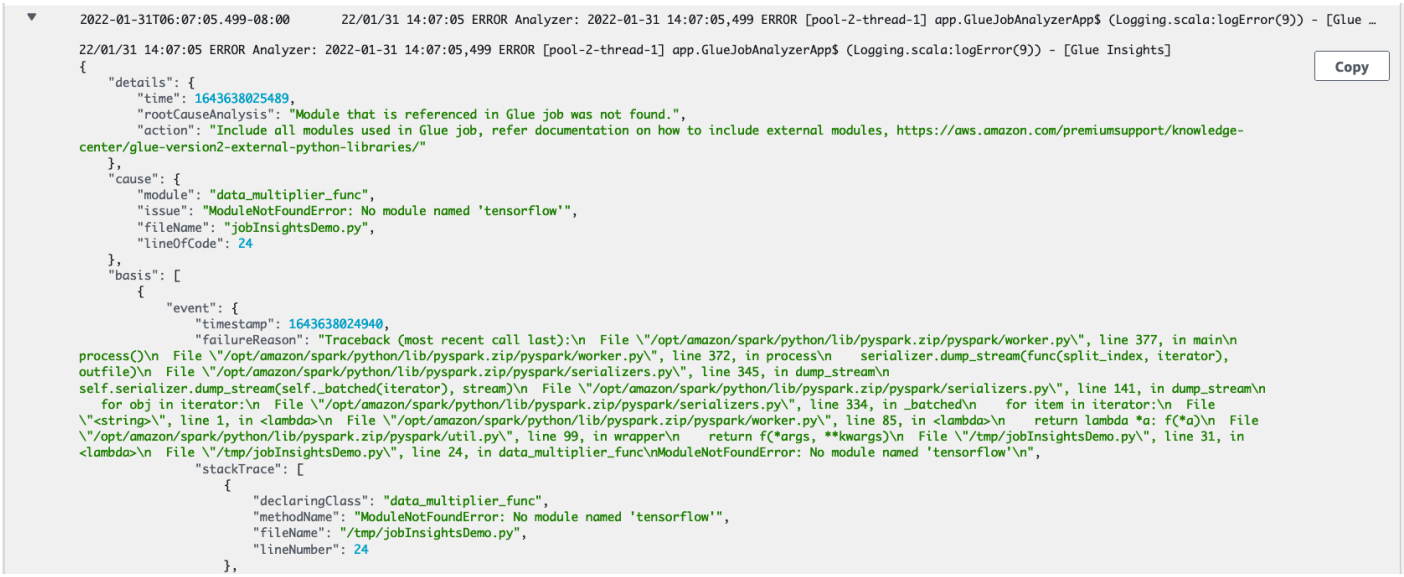
A mensagem é ambígua e limita sua experiência de depuração. Nesse caso, esse recurso fornece informações adicionais em dois fluxos de CloudWatch log:

1. O fluxo de log do `job-insights-rca-driver`:
   - Eventos de exceção: este fluxo de log fornece os eventos de exceção do Spark relacionados à falha coletada do driver do Spark e de diferentes operadores distribuídos. Esses eventos ajudam a entender a propagação ordenada por tempo da exceção à medida que o código defeituoso é executado em tarefas, executores e estágios do Spark distribuídos pelos operadores do AWS Glue.
   - Números de linha: este fluxo de log identifica a linha 21, que fez a chamada para importar o módulo Python ausente que causou a falha; ele também identifica a linha 24, a chamada para a ação `collect()` do Spark, como a última linha executada em seu script.



2. O fluxo de log do `job-insights-rule-driver`:

   - Causa raiz e recomendação: além do número da linha e do número da última linha executada para a falha no script, esse fluxo de log mostra a análise da causa raiz e a recomendação para que você siga o doc do AWS Glue e configure os parâmetros de trabalho necessários para usar um módulo Python adicional em seu trabalho do AWS Glue.
   - Evento base: este fluxo de log também mostra o evento de exceção do Spark que foi avaliado com a regra definida pelo serviço para inferir a causa raiz e fornecer uma recomendação.



Monitorar com o Amazon CloudWatch

É possível monitorar o AWS Glue usando o Amazon CloudWatch, que coleta e processa dados brutos do AWS Glue e os transforma em métricas legíveis quase em tempo real. Essas estatísticas são registradas por um período de duas semanas, para que você possa acessar informações históricas para obter uma perspectiva melhor sobre a performance do seu serviço ou da sua

aplicação Web. Por padrão, os dados de métricas do AWS Glue são enviados para o CloudWatch automaticamente. Para obter mais informações, consulte O que é o Amazon CloudWatch? no Manual do usuário do Amazon CloudWatch e em AWS Glue métricas.

Log contínuo

O AWS Glue também oferece suporte a registro contínuo em tempo real para tarefas do AWS Glue. Quando o registro em log contínuo está habilitado para um trabalho, é possível visualizar os logs em tempo real no console do AWS Glue ou no painel do console do CloudWatch. Para ter mais informações, consulte Registro contínuo para trabalhos do AWS Glue.

Métricas de observabilidade

Quando as Métricas de observabilidade do trabalho estão ativadas, Amazon CloudWatch métricas adicionais são geradas quando o trabalho é executado. Use métricas de observabilidade do AWS Glue para gerar insights sobre o que está acontecendo dentro do seu AWS Glue para melhorar a triagem e a análise de problemas.

Tópicos

- Monitorar o AWS Glue usando métricas do Amazon CloudWatch
- Configurar alarmes do Amazon CloudWatch nos perfis de trabalho do AWS Glue
- Registro contínuo para trabalhos do AWS Glue
- Monitoramento com métricas de observabilidade do AWS Glue

Monitorar o AWS Glue usando métricas do Amazon CloudWatch

É possível criar perfis e monitorar operações do AWS Glue usando o criador de perfis de tarefas do AWS Glue. Ele coleta e processa dados brutos de trabalhos do AWS Glue e os transforma em métricas legíveis, quase em tempo real, armazenadas no Amazon CloudWatch. Essas estatísticas são retidas e agregadas no CloudWatch, para que você possa acessar informações históricas e obter uma perspectiva melhor sobre como sua aplicação está se saindo.

> **Note**
>
> Você pode incorrer em cobranças adicionais ao habilitar que métricas de trabalho e métricas personalizadas do CloudWatch sejam criadas. Para obter mais informações, consulte Preço do Amazon CloudWatch.

Visão geral das métricas do AWS Glue

Quando você interage com o AWS Glue, ele envia métricas ao CloudWatch. É possível visualizar essas métricas usando o console do AWS Glue (o método de preferência), o painel do console do CloudWatch ou a AWS Command Line Interface (AWS CLI).

Para visualizar as métricas usando o painel do console do AWS Glue

Você pode visualizar gráficos de métricas detalhados ou resumidos de uma tarefa, ou gráficos detalhadas da execução de uma tarefa.

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Monitoramento de execução de trabalho.
3. Em Execuções de trabalhos, escolha Ações para interromper um trabalho que está em execução no momento, exibir um trabalho ou retroceder o marcador de trabalho.
4. Selecione um trabalho e escolha Exibir detalhes da execução para ver informações adicionais sobre a execução do trabalho.

Para exibir métricas usando o painel do console do CloudWatch

As métricas são agrupadas primeiro pelo namespace do serviço e, em seguida, por várias combinações de dimensão dentro de cada namespace.

1. Abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Métricas.
3. Selecione o namespace Glue.

Para visualizar métricas usando a AWS CLI

- Em um prompt de comando, use o seguinte comando.

```
aws cloudwatch list-metrics --namespace Glue
```

O AWS Glue relata métricas para o CloudWatch a cada 30 segundos, e os painéis de métricas do CloudWatch são configurados para exibi-las a cada minuto. As métricas do AWS Glue representam valores do delta a partir dos valores relatados anteriormente. Quando apropriado, os painéis de

métricas agregam (somam) os valores de 30 segundos para obter um valor para o último minuto inteiro.

Comportamento de métricas do AWS Glue para trabalhos do Spark

As métricas do AWS Glue são ativadas na inicialização de um `GlueContext` em um script e geralmente são atualizadas somente no final de um trabalho do Apache Spark. Eles representam os valores agregados em todas as tarefas do Spark concluídas até agora.

No entanto, as métricas do Spark que o AWS Glue transfere para o CloudWatch são geralmente valores absolutos que representam o estado atual no momento em que elas são relatadas. O AWS Glue as informa ao CloudWatch a cada 30 segundos, e os painéis de métricas geralmente mostram a média em todos os pontos de dados recebidos no último minuto.

Os nomes de métricas do AWS Glue são todos precedidos por um dos seguintes tipos de prefixo:

- `glue.driver.`: métricas cujos nomes começam com esse prefixo representam métricas do AWS Glue agregadas em todos os executores no driver do Spark ou métricas do Spark correspondentes ao driver do Spark.
- `glue.`executorId`.`: o executorId é o número de um executor específico do Spark. Ele corresponde ao executores listados nos logs.
- `glue.ALL.` - Métricas cujos nomes começam com esse prefixo agregam valor a todos os executores do Spark.

AWS Glue métricas

O AWS Glue traça o perfil e envia as seguintes métricas para o CloudWatch a cada 30 segundos, e o painel do console do AWS Glue as relata uma vez por minuto:

Métrica	Descrição
`glue.driver.aggregate.bytesRead`	O número de bytes lidos de todas as fontes de dados por todas as tarefas do Spark concluídas em execução em todos os executores. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem).

Métrica	Descrição
	Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. Unidade: bytes Pode ser usada para monitorar: • Bytes lidos. • Progresso do trabalho. • Fontes de dados JDBC. • Problemas com marcadores do trabalho. • Variação entre execuções do trabalho. Essa métrica pode ser usada da mesma maneira que a métrica `glue.ALL.s3.filesystem.read_bytes` , com a diferença de que essa métrica é atualizada no final de uma tarefa do Spark e também captura fontes de dados que não sejam do S3.

Métrica	Descrição
`glue.driver.aggregate.elapsedTime`	O tempo de ETL decorrido em milissegundos (não inclui os tempos de bootstrap do trabalho). Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem). Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. Unidade: milissegundos Pode ser usado para determinar quanto tempo leva em média para uma execução de trabalho ser executada. Algumas maneiras de usar os dados: • Ajustar alarmes para retardatários. • Medir a variação entre execuções do trabalho.

Métrica	Descrição
`glue.driver.aggregate.numCompletedStages`	O número de etapas concluídas no trabalho. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem). Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. Unidade: Contagem Pode ser usada para monitorar: • Progresso do trabalho. • Cronograma por etapa de execução de trabalhos, quando correlacionado com outras métricas. Algumas maneiras de usar os dados: • Identificar etapas exigentes na execução de um trabalho. • Definir alarmes para picos correlacionados (etapas exigentes) em execuções de trabalhos.

Métrica	Descrição
`glue.driver.aggregate.numCompletedTasks`	O número de tarefas concluídas no trabalho. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem). Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. Unidade: Contagem Pode ser usada para monitorar: • Progresso do trabalho. • Paralelismo dentro de uma etapa.

Métrica	Descrição
`glue.driver.aggregate.numFailedTasks`	O número de tarefas que falharam. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem). Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. Unidade: Contagem Pode ser usada para monitorar: • Anormalidades de dados que fazem com que as tarefas de trabalhos falhem. • Anormalidades de cluster que fazem com que as tarefas de trabalhos falhem. • Anormalidades de script que fazem com que as tarefas de trabalhos falhem. Os dados podem ser usados para definir alarmes para falhas aumentadas que possam sugerir anormalidades em dados, cluster ou scripts.

Métrica	Descrição
`glue.driver.aggregate.numKilledTasks`	O número de tarefas encerradas. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem). Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. Unidade: Contagem Pode ser usada para monitorar: • Anormalidades na distorção de dados que resultam em exceções (OOMs) que encerram tarefas. • Anormalidades no script que resultam em exceções (OOMs) que encerram tarefas. Algumas maneiras de usar os dados: • Definir alarmes para falhas aumentadas indicando anormalidades dos dados. • Definir alarmes para falhas aumentadas indicando anormalidades do cluster. • Definir alarmes para falhas aumentadas indicando anormalidades dos scripts.

Métrica	Descrição
`glue.driver.aggregate.recordsRead`	O número de registros lidos de todas as fontes de dados por todas as tarefas do Spark concluídas em execução em todos os executores. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem). Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. Unidade: Contagem Pode ser usada para monitorar: • Registros lidos. • Progresso do trabalho. • Fontes de dados JDBC. • Problemas com marcadores do trabalho. • Distorções em execuções de trabalhos ao longo dos dias. Essa métrica pode ser usada da mesma maneira que a métrica `glue.ALL.s3.filesystem.read_bytes`, com a diferença de que essa métrica é atualizada no final de uma tarefa do Spark.

Métrica	Descrição
`glue.driver.aggregate.shuffleBytesWritten`	O número de bytes gravados por todos os executores para gerar a ordem aleatória de dados entre eles desde o relatório anterior (agregado pelo painel de métricas do AWS Glue como o número de bytes gravados para esse fim durante o minuto anterior). Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem). Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. Unidade: bytes Pode ser usado para monitorar: ordem aleatória de dados em trabalhos (junções grandes, groupBy, repartição, união). Algumas maneiras de usar os dados: • Reparticionar ou descompactar arquivos de entrada grandes antes de processamento adicional. • Reparticionar dados de forma mais uniforme para evitar teclas de atalho. • Pré-filtrar dados antes de junções ou operações de groupBy.

Métrica	Descrição
`glue.driver.aggregate.shuffleLocalBytesRead`	O número de bytes lidos por todos os executores para gerar a ordem aleatória dos dados entre eles desde o relatório anterior (agregado pelo painel de métricas do AWS Glue como o número de bytes lidos para esse fim durante o minuto anterior). Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem). Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. Unidade: bytes Pode ser usado para monitorar: ordem aleatória de dados em trabalhos (junções grandes, groupBy, repartição, união). Algumas maneiras de usar os dados: • Reparticionar ou descompactar arquivos de entrada grandes antes de processamento adicional. • Reparticionar dados de forma mais uniforme usando teclas de atalho. • Pré-filtrar dados antes de junções ou operações de groupBy.

Métrica	Descrição
`glue.driver.BlockManager.disk.diskSpaceUsed_MB`	O número de megabytes de espaço em disco usado em todos os executores. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (medida). Estatística válida: média. Essa é uma métrica do Spark, relatada como um valor absoluto. Unidade: megabytes Pode ser usada para monitorar: • Espaço em disco usado para blocos que represent am partições RDD armazenadas em cache. • Espaço em disco usado para blocos que represent am saídas em ordem aleatória intermediárias. • Espaço em disco usado para blocos que represent am emissões. Algumas maneiras de usar os dados: • Identificar falhas de trabalho pelo aumento do uso do disco. • Identificar partições grandes que resultam em despejo ou embaralhamento. • Aumentar a capacidade da DPU provisionada para corrigir esses problemas.

Métrica	Descrição
`glue.driver.ExecutorAllocationManager.executors.numberAllExecutors`	O número de executores ativamente executando trabalhos. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (medida). Estatística válida: média. Essa é uma métrica do Spark, relatada como um valor absoluto. Unidade: Contagem Pode ser usada para monitorar: • Atividade do trabalho. • Executadores atrasados (com alguns executores em execução apenas) • Paralelismo atual no nível do executor. Algumas maneiras de usar os dados: • Reparticionar ou descompactar arquivos de entrada grandes de antemão, se o cluster estiver subutilizado. • Identificar atrasos na etapa ou na execução do trabalho por cenários atrasados. • • Compare com numberMaxNeededExecutors para entender o backlog para provisionamento de mais DPUs.

Métrica	Descrição
`glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors`	O número máximo de executores de trabalho (em execução ativa e pendentes) necessários para satisfazer a carga atual. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (medida). Estatísticas válidas: máximo. Essa é uma métrica do Spark, relatada como um valor absoluto. Unidade: Contagem Pode ser usada para monitorar: • Atividade do trabalho. • Paralelismo atual no nível do executor e backlog de tarefas pendentes ainda não agendadas por causa de executores indisponíveis devido à capacidade da DPU ou executores encerrados/falhados. Algumas maneiras de usar os dados: • Identificar pendência/backlog da fila de agendamento. • Identificar atrasos na etapa ou na execução do trabalho por cenários atrasados. • Comparar com numberAllExecutors para entender o backlog para provisionamento de mais DPUs. • Aumentar a capacidade da DPU provisionada para corrigir o backlog pendente do executor.

Métrica	Descrição

`glue.driver.jvm.heap.usage`
 	
`glue.executorId.jvm.heap.usage`
 	
`glue.ALL.jvm.heap.usage`
	A fração de memória usada pelo heap da JVM para este driver (escala: 0 a 1) pelo driver, executor identificado por executorId ou TODOS os executore s.
	
	Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (medida).
	
	Estatística válida: média. Essa é uma métrica do Spark, relatada como um valor absoluto.
	
	Unidade: porcentagem
	
	Pode ser usada para monitorar:
	
	• Condições de falta de memória (OOM) do driver usando `glue.driver.jvm.heap.usage` .
	• Condições de falta de memória (OOM) do executor usando `glue.ALL.jvm.heap.usage` .
	
	Algumas maneiras de usar os dados:
	
	• Identificar os IDs e etapas do executor que consomem memória.
	• Identificar os IDs e etapas do executor atrasado.
	• Identificar uma condição de falta de memória (OOM) do driver.
	• Identificar uma condição de falta de memória (OOM) do executor e obter o ID de executor correspondente, para obter um rastreamento de pilha do log do executor.
	• Identificar arquivos ou partições que podem ter distorção de dados resultando em atrasos ou condições de falta de memória (OOMs).

Métrica	Descrição

`glue.driver.jvm.heap.used`
 	
`glue.executorId.jvm.heap.used`
 	
`glue.ALL.jvm.heap.used`
	O número de bytes de memória usados pelo heap da JVM para o driver, o executor identificado por executorId ou TODOS os executores.
	
	Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (medida).
	
	Estatística válida: média. Essa é uma métrica do Spark, relatada como um valor absoluto.
	
	Unidade: bytes
	
	Pode ser usada para monitorar:
	
	• Condições de falta de memória (OOM) do driver.
	• Condições de falta de memória (OOM) do executor.
	
	Algumas maneiras de usar os dados:
	
	• Identificar os IDs e etapas do executor que consomem memória.
	• Identificar os IDs e etapas do executor atrasado.
	• Identificar uma condição de falta de memória (OOM) do driver.
	• Identificar uma condição de falta de memória (OOM) do executor e obter o ID de executor correspondente, para obter um rastreamento de pilha do log do executor.
	• Identificar arquivos ou partições que podem ter distorção de dados resultando em atrasos ou condições de falta de memória (OOMs).

Métrica	Descrição

`glue.driver.s3.filesystem.read_bytes`
 	
`glue.executorId.s3.filesystem.read_bytes`
 	
`glue.ALL.s3.filesystem.read_bytes`
	O número de bytes lidos do Amazon S3 pelo driver, um executor identificado por executorId ou TODOS os executores desde o relatório anterior (agregado pelo painel de métricas do AWS Glue como o número de bytes lidos durante o minuto anterior).
	
	Dimensões válidas: `JobName`, `JobRunId` e `Type` (medida).
	
	Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. A área sob a curva no painel de métricas do AWS Glue pode ser usada para comparar visualmente os bytes lidos por duas execuções de trabalho diferentes.
	
	Unidade: Bytes.
	
	Pode ser usada para monitorar:
	
	• Movimentação de dados de ETL.
	• Progresso do trabalho.
	• Problemas com marcadores de trabalhos (dados processados, reprocessados e ignorados).
	• Comparação de leituras com a taxa de ingestão de fontes de dados externas.
	• Variação entre execuções do trabalho.
	
	Os dados resultantes podem ser usados para:
	
	• Planejar a capacidade da DPU.
	• Configurar alarmes para grandes picos ou quedas de dados lidos para execuções e etapas de trabalhos.

Métrica	Descrição

`glue.driver.s3.filesystem.write_bytes`
 	
`glue.executorId.s3.filesystem.write_bytes`
 	
`glue.ALL.s3.filesystem.write_bytes`
	O número de bytes gravados no Amazon S3 pelo driver, um executor identificado por executorId ou TODOS os executores desde o relatório anterior (agregado pelo painel de métricas do AWS Glue como o número de bytes gravados durante o minuto anterior).
	
	Dimensões válidas: `JobName`, `JobRunId` e `Type` (medida).
	
	Estatística válida: SUM (SOMA). Essa métrica é um valor delta do último valor relatado, portanto, no painel de métricas do AWS Glue, uma estatística SUM (SOMA) é usada para agregação. A área sob a curva no painel de métricas do AWS Glue pode ser usada para comparar visualmente os bytes gravados por duas execuções de trabalho diferentes.
	
	Unidade: bytes
	
	Pode ser usada para monitorar:
	
	• Movimentação de dados de ETL.
	• Progresso do trabalho.
	• Problemas com marcadores de trabalhos (dados processados, reprocessados e ignorados).
	• Comparação de leituras com a taxa de ingestão de fontes de dados externas.
	• Variação entre execuções do trabalho.
	
	Algumas maneiras de usar os dados:
	
	• Planejar a capacidade da DPU.

Métrica	Descrição
	• Configurar alarmes para grandes picos ou quedas de dados lidos para execuções e etapas de trabalhos.
`glue.driver.streaming.numRecords`	O número de registros que são recebidos em um microlote. Essa métrica só está disponível para os trabalhos de transmissão do AWS Glue com o AWS Glue versão 2.0 e posterior. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem). Estatísticas válidas: soma , máximo, mínimo, média, porcentagem Unidade: Contagem Pode ser usada para monitorar: • Registros lidos. • Progresso do trabalho.

Métrica	Descrição
`glue.driver.streaming.batchProcessingTimeInMs`	O tempo necessário para processar os lotes em milissegundos. Essa métrica só está disponível para os trabalhos de transmissão do AWS Glue com o AWS Glue versão 2.0 e posterior. Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (contagem). Estatísticas válidas: soma , máximo, mínimo, média, porcentagem Unidade: Contagem Pode ser usada para monitorar: • Progresso do trabalho. • Performance do script.

Métrica	Descrição

`glue.driver.system.cpuSystemLoad`
 	
`glue.executorId.system.cpuSystemLoad`
 	
`glue.ALL.system.cpuSystemLoad`
	A fração de carga de sistema da CPU usada (escala: 0 a 1) pelo driver, um executor identificado por executorId ou TODOS os executores.
	
	Dimensões válidas: `JobName` (o nome do trabalho do AWS Glue), `JobRunId` (o ID de JobRun. ou `ALL`) e `Type` (medida).
	
	Estatística válida: média. Essa métrica é relatada como um valor absoluto.
	
	Unidade: porcentagem
	
	Pode ser usada para monitorar:
	
	• Carga de CPU do driver.
	• Carga de CPU do executor.
	• Detecção de executores ou etapas vinculados à CPU ou E/S em um trabalho.
	
	Algumas maneiras de usar os dados:
	
	• Planejar a capacidade da DPU junto com métricas de E/S (bytes de leitura/ordem aleatória de de bytes, paralelismo de tarefas) e a métrica de número máximo de executores necessários.
	• Identificar a relação entre vinculção à CPU e a E/S. Isso permite reparticionar e aumentar a capacidade provisionada para trabalhos de longa execução com conjuntos de dados divisíveis com menor utilização da CPU.

Dimensões para métricas do AWS Glue

As métricas do AWS Glue usam namespace do AWS Glue e fornecem métricas para as seguintes dimensões:

Dimensão	Descrição
`JobName`	Esta dimensão filtra as métricas de todas as execuções de trabalho de um trabalho do AWS Glue.
`JobRunId`	Esta dimensão filtra as métricas de um trabalho do AWS Glue específico executado por um ID JobRun ou `ALL`.
`Type`	Esta dimensão filtra as métricas por `count` (um número agregado) ou `gauge` (um valor em um determinado ponto no tempo).

Para mais informações, consulte o Guia do usuário do Amazon CloudWatch.

Configurar alarmes do Amazon CloudWatch nos perfis de trabalho do AWS Glue

As métricas do AWS Glue também estão disponíveis no Amazon CloudWatch. Você pode configurar alarmes em qualquer métrica do AWS Glue para tarefas agendadas.

Alguns cenários comuns para a configuração de alarmes são os seguintes:

- Trabalhos sem memória (OOM): defina um alarme quando o uso de memória exceder a média normal para o driver ou um executor para um trabalho do AWS Glue.
- Executores difíceis: defina um alarme quando o número de executores cair abaixo de um determinado limite por um grande período em um trabalho do AWS Glue.
- Backlog ou reprocessamento de dados: compare as métricas de trabalhos individuais em um fluxo de trabalho usando uma expressão matemática do CloudWatch. Em seguida, você pode acionar um alarme no valor da expressão resultante (como a proporção de bytes gravados por um trabalho e bytes lidos por um trabalho seguinte).

Para obter instruções detalhadas sobre como definir alarmes, consulte Criar ou editar um alarme do CloudWatch no Manual do usuário do Amazon CloudWatch Events.

Para cenários de monitoramento e depuração usando o CloudWatch, consulte Monitoramento e depuração de trabalho.

Registro contínuo para trabalhos do AWS Glue

O AWS Glue fornece o registro contínuo e em tempo real para tarefas do AWS Glue. É possível visualizar logs de trabalhos do Apache Spark em tempo real no Amazon CloudWatch, incluindo logs de driver, logs de executor e uma barra de progresso de trabalhos do Apache Spark. A visualização de logs em tempo real fornece uma perspectiva melhor sobre a execução da tarefa.

Quando você inicia um trabalho do AWS Glue, ele envia as informações de registro em tempo real para o CloudWatch (a cada cinco segundos e antes de cada término do executor) depois que a aplicação do Spark começa a ser executada. É possível visualizar os logs no console do AWS Glue ou no painel do console do CloudWatch.

O recurso de registro contínuo inclui os seguintes recursos:

- Registro em log contínuo
- Um registrador de script personalizado para registar em log mensagens específicas do aplicativo
- Uma barra de progresso do console para rastrear o status da execução da tarefa atual do AWS Glue

Para obter informações sobre como o registro em log contínuo é suportado no AWS Glue versão 2.0, consulte Executar trabalhos de ETL do Spark com tempos de inicialização reduzidos.

Você pode restringir o acesso a grupos de log do CloudWatch ou transmissões para funções do IAM para ler os logs. Para obter mais detalhes sobre como restringir o acesso, consulte Uso de políticas baseadas em identidade (políticas do IAM) para o CloudWatch Logs na documentação do CloudWatch.

> **Note**
>
> Você pode incorrer em cobranças adicionais ao habilitar que o registro contínuo e os eventos de log adicionais do CloudWatch sejam criados. Para obter mais informações, consulte Preço do Amazon CloudWatch.

Tópicos

- Habilitar o registro contínuo para trabalhos do AWS Glue
- Visualizar o registro contínuo para trabalhos do AWS Glue

Habilitar o registro contínuo para trabalhos do AWS Glue

É possível habilitar o registro contínuo usando o console do AWS Glue ou por meio da AWS Command Line Interface (AWS CLI).

É possível habilitar o registro em log contínuo ao criar um trabalho ou editar um trabalho existente ou habilitá-lo via AWS CLI.

Também é possível especificar opções de configuração personalizadas, como o nome do grupo de logs do Amazon CloudWatch, o prefixo da transmissão de logs do CloudWatch antes do ID de driver/ID de executor da execução de trabalho do AWS Glue e o padrão de conversão de logs para mensagens de log. Essas configurações ajudam a definir logs agregados em grupos de logs personalizados do CloudWatch com diferentes políticas de expiração e a analisá-los em mais detalhes com prefixos de fluxo de logs e padrões de conversões personalizados.

Tópicos

- Usando a AWS Management Console
- Registrar em log mensagens específicas da aplicação usando o registrador de script personalizado
- Habilitar a barra de progresso para exibir o progresso do trabalho
- Configuração de segurança com registro contínuo

Usando a AWS Management Console

Siga estas etapas para usar o console para habilitar o registro contínuo ao criar ou editar uma tarefa do AWS Glue.

Para criar uma tarefa do AWS Glue com registro contínuo

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Trabalhos de ETL.
3. Escolha ETL visual.
4. Na guia Detalhes do trabalho, expanda a seção Propriedades avançadas.

5. Em Registro em log contínuo, selecione Ativar logs no CloudWatch.

Para habilitar o registro contínuo para um tarefa existente do AWS Glue

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Tarefas.
3. Escolha uma tarefa existente na lista Jobs (Tarefas).
4. Selecione Action (Ação), Edit job (Editar tarefa).
5. Na guia Detalhes do trabalho, expanda a seção Propriedades avançadas.
6. Em Registro em log contínuo, selecione Ativar logs no CloudWatch.

Usando a AWS CLI

Para habilitar o registro contínuo, transmita parâmetros de tarefa para uma tarefa do AWS Glue. Transmita os parâmetros de trabalho especiais a seguir semelhantes a outros parâmetros de trabalho do AWS Glue. Para ter mais informações, consulte Parâmetros de trabalho do AWS Glue.

```
'--enable-continuous-cloudwatch-log': 'true'
```

É possível especificar um nome de grupo de logs personalizado do Amazon CloudWatch. Se não for especificado, o nome do grupo de logs padrão será `/aws-glue/jobs/logs-v2/`.

```
'--continuous-log-logGroup': 'custom_log_group_name'
```

É possível especificar um prefixo de transmissão de logs personalizado do Amazon CloudWatch. Se não for especificado, o prefixo do fluxo de logs padrão será o ID de execução do trabalho.

```
'--continuous-log-logStreamPrefix': 'custom_log_stream_prefix'
```

É possível especificar um padrão contínuo e personalizado de conversão de logs. Se não for especificado, o padrão de conversão padrão será `%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n`. Observe que o padrão de conversão só se aplica a logs de driver e logs de executor. Isso não afeta a barra de progresso do AWS Glue.

```
'--continuous-log-conversionPattern': 'custom_log_conversion_pattern'
```

Registrar em log mensagens específicas da aplicação usando o registrador de script personalizado

É possível usar o registrador do AWS Glue para registrar qualquer mensagem específica do aplicativo no script que foi enviado em tempo real ao stream de log do driver.

O exemplo a seguir mostra um script 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")
```

O exemplo a seguir mostra um script 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")
  }
}
```

Habilitar a barra de progresso para exibir o progresso do trabalho

O AWS Glue fornece uma barra de progresso em tempo real no stream de log `JOB_RUN_ID-progress-bar` para verificar o status de execução da tarefa do AWS Glue. No momento, ele oferece suporte somente a tarefas que inicializam o `glueContext`. Se você executar uma tarefa pura do Spark sem inicializar o `glueContext`, a barra de progresso do AWS Glue não será exibida.

A barra de progresso mostra a atualização de progresso a seguir a cada 5 segundos.

```
Stage Number (Stage Name): > (numCompletedTasks + numActiveTasks) /
 totalNumOfTasksInThisStage]
```

Configuração de segurança com registro contínuo

Se uma configuração de segurança estiver habilitada para logs do CloudWatch, o AWS Glue criará um grupo de logs chamado da seguinte forma para logs contínuos:

```
<Log-Group-Name>-<Security-Configuration-Name>
```

Os grupos de log padrão e personalizado serão os seguintes:

- O grupo de logs contínuos padrão será `/aws-glue/jobs/logs-v2-`*`<Security-Configuration-Name>`*
- O grupo de logs contínuos personalizado será *`<custom-log-group-name>`*`-`*`<Security-Configuration-Name>`*

É necessário adicionar `logs:AssociateKmsKey` a suas permissões de função do IAM, se você habilitar uma configuração de segurança com o CloudWatch Logs. Se essa permissão não for incluída, o registro em log contínuo será desabilitado. Além disso, para configurar a criptografia do CloudWatch Logs, siga as instruções em Criptografar dados de log no CloudWatch Logs usando o AWS Key Management Service no Manual do usuário do Amazon CloudWatch Logs.

Para obter mais informações sobre como criar configurações de segurança, consulte Trabalhar com configurações de segurança no console do AWS Glue.

Visualizar o registro contínuo para trabalhos do AWS Glue

É possível visualizar logs em tempo real usando o console do AWS Glue ou o console do Amazon CloudWatch.

Para visualizar logs em tempo real usando o painel do console do AWS Glue

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Tarefas.
3. Adicione ou inicie uma tarefa existente. Selecione Action (Ação), Run job (Executar tarefa).

   Ao começar a executar uma tarefa, você navega até uma página que contém informações sobre a tarefa em execução:

   - A guia Logs mostra os logs do aplicativo agregados mais antigos.

- A guia Continuous logging (Registro contínuo) mostra uma barra de progresso em tempo real quando a tarefa está sendo executada com o `glueContext` inicializado.
- A guia Continuous logging (Registro contínuo) também contém os Driver logs (Logs de driver), que capturam logs de driver do Apache Spark em tempo real, além de logs de aplicativo do script registrado usando o registrador de aplicativo do AWS Glue quando a tarefa está em execução.

4. Para tarefas mais antigas, também é possível visualizar logs em tempo real na exibição Job History (Histórico de tarefas) selecionando Logs. Essa ação leva você até o console do CloudWatch, que mostra todas as transmissões de log da barra de progresso, do executor e do driver do Spark para essa execução de trabalho.

Para visualizar logs em tempo real usando o painel do console do CloudWatch

1. Abra o console CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Log.
3. Selecione o grupo de logs /aws-glue/jobs/logs-v2/.
4. Na caixa Filter (Filtro), cole o ID da execução da tarefa.

   É possível visualizar os logs de driver, logs de executor e a barra de progresso (se estiver usando o Standard filter (Filtro padrão)).

Monitoramento com métricas de observabilidade do AWS Glue

> **Note**
>
> As métricas de observabilidade do AWS Glue estão disponíveis no AWS Glue 4.0 e versões posteriores.

Use métricas de observabilidade do AWS Glue para gerar insights sobre o que está acontecendo dentro do seu AWS Glue para trabalhos do Apache Sparks para melhorar a triagem e a análise de problemas. As métricas de observabilidade são visualizadas por meio de painéis do Amazon CloudWatch e podem ser usadas para ajudar a realizar a análise da causa raiz de erros e para diagnosticar gargalos de performance. É possível reduzir o tempo gasto na depuração de problemas em escala para que você possa se concentrar em resolver problemas com mais rapidez e eficiência.

A observabilidade do AWS Glue fornece métricas do Amazon CloudWatch categorizadas nos quatro grupos a seguir:

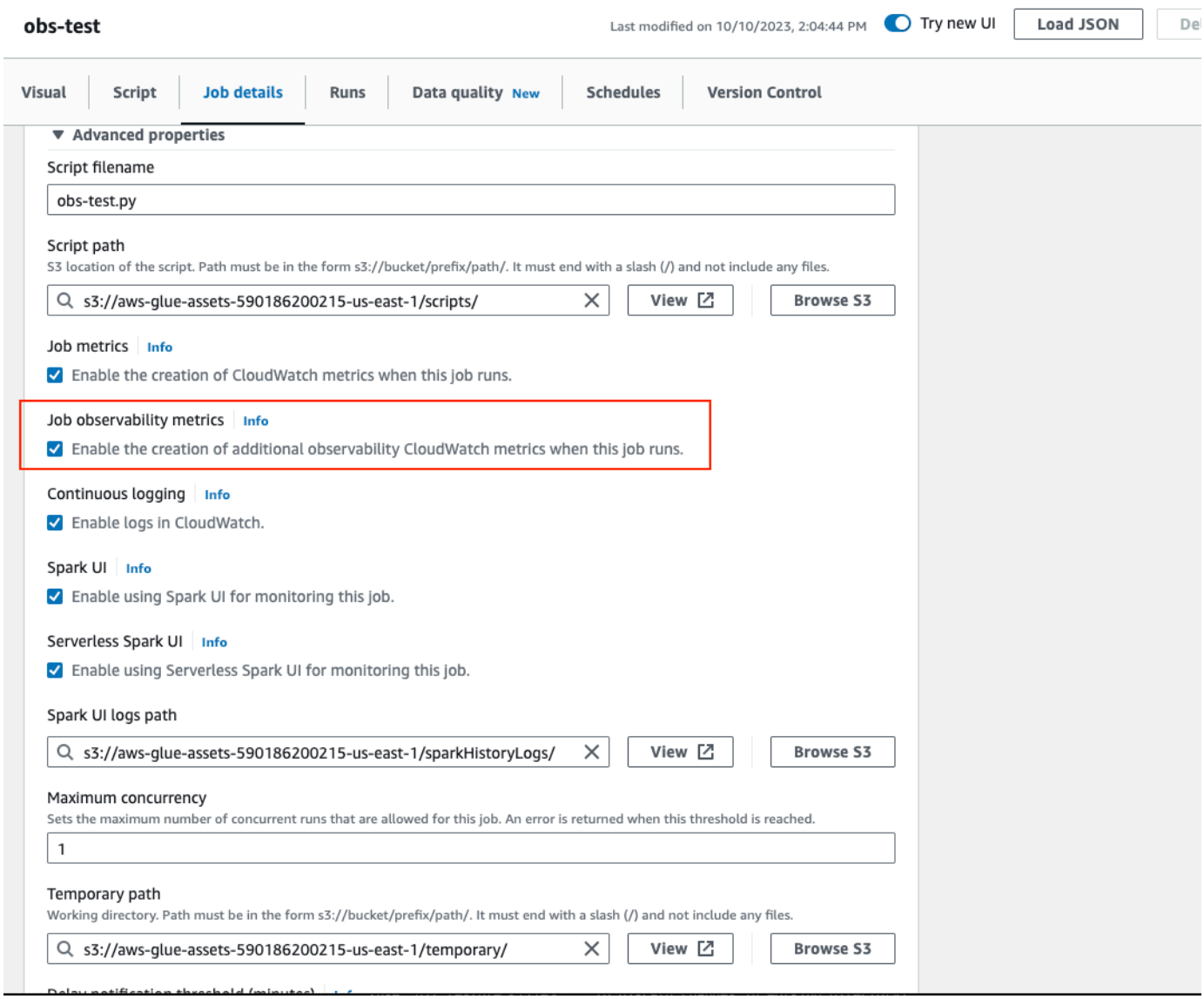
- Confiabilidade (ou seja, classes de erros): identifique facilmente os motivos de falha mais comuns em um determinado intervalo de tempo que você talvez queira resolver.
- Performance (ou seja, assimetria): identifique um gargalo de performance e aplique técnicas de ajuste. Por exemplo, quando você enfrenta uma degradação de performance causada pela assimetria de um trabalho, convém ativar a execução adaptativa de consultas do Spark e ajustar o limite de junção assimétrica.
- Throughput (ou seja, throughput por fonte/coletor): monitore as tendências de leituras e gravações de dados. Você também pode configurar alarmes do Amazon CloudWatch para anomalias.
- Utilização de recursos (ou seja, operadores, memória e utilização de disco): encontre de forma eficiente os trabalhos com baixa utilização de capacidade. Talvez você queira habilitar o ajuste de escala automático do AWS Glue para esses trabalhos.

Conceitos básicos das métricas de observabilidade do AWS Glue

> **Note**
>
> As métricas novas são habilitadas por padrão no console do AWS Glue Studio.

Para configurar métricas de observabilidade no AWS Glue Studio:

1. Faça login no console do AWS Glue e escolha Trabalhos ETL no menu do console.
2. Escolha um trabalho clicando no nome do trabalho na seção Seus trabalhos.
3. Escolha a guia Job details (Detalhes do trabalho).
4. Role até a parte inferior e escolha Propriedades avançadas e, em seguida, Métricas de observabilidade do trabalho.



Para habilitar as métricas de observabilidade do AWS Glue usando a AWS CLI:

- Adicione ao mapa `--default-arguments` o seguinte par chave-valor no arquivo JSON de entrada:

```
--enable-observability-metrics, true
```

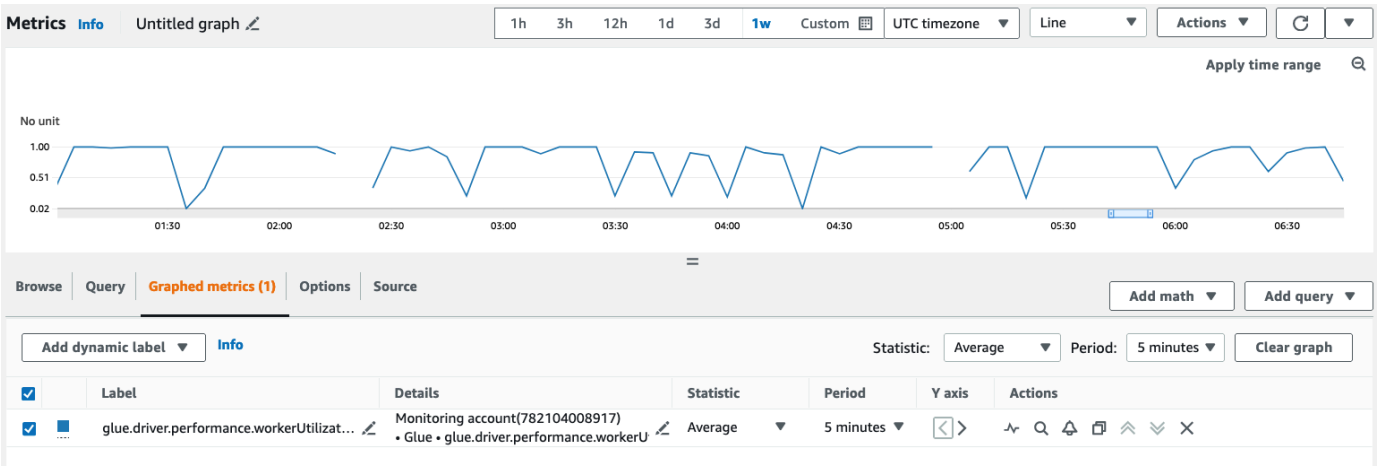
Usar a observabilidade do AWS Glue

Como as métricas de observabilidade do AWS Glue são fornecidas por meio do Amazon CloudWatch, você pode usar o console do Amazon CloudWatch, a AWS CLI, o SDK ou a API para consultar os pontos de dados das métricas de observabilidade. Consulte Como usar a observabilidade do Glue para monitorar a utilização de recursos para reduzir custos para ver um exemplo de caso de uso de métricas de observabilidade do AWS Glue.

Usar a observabilidade do AWS Glue no console do Amazon CloudWatch

Para consultar e visualizar métricas no console do Amazon CloudWatch:

1. Abra o console do Amazon CloudWatch e escolha Todas as métricas.
2. Em Namespaces personalizados, escolha AWS Glue.
3. Escolha Métricas de observabilidade do trabalho, Métricas de observabilidade por fonte ou Métricas de observabilidade por coletor.
4. Pesquise o nome da métrica específica, o nome do trabalho, o ID da execução do trabalho e selecione-os.
5. Na guia Métricas representadas graficamente, configure a estatística, o período e outras opções preferidas.



Para consultar uma métrica de observabilidade usando a AWS CLI:

1. Crie um arquivo JSON de definição métrica e substitua `your-Glue-job-name` e `your-Glue-job-run-id` pelos seus.

```
$ 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. Execute o 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 observabilidade

A observabilidade do AWS Glue traça o perfil e envia as seguintes métricas do Amazon CloudWatch a cada 30 segundos. Algumas dessas métricas podem ser visíveis na página de monitoramento de execução de trabalhos do AWS Glue Studio.

Métrica	Descrição	Categoria
glue.driver.skewness.stage	Categoria métrica: job_performance Distorção da assimetria dos estágios do Spark: essa métrica captura a assimetria da execução, que pode ser causada pela assimetria dos dados de entrada ou por uma transformação (p. ex., junção assimétrica). Os valores dessa métrica estão na faixa de [0, infinito[, em que 0 significa que a razão entre o tempo máximo e a mediana de execução das tarefas, entre todas as tarefas no estágio, é menor que um determinado	job_performance

Métrica	Descrição	Categoria
	fator de distorção do estágio. O fator de assimetria de estágio padrão é `5` e pode ser sobrescrito por meio do spark conf: spark.metrics.conf .driver.source.glue.jobPerf ormance.skewnessFactor Um valor de assimetria de estágio de 1 significa que a proporção é o dobro do fator de assimetria do estágio. O valor de assimetria do estágio é atualizado a cada 30 segundos para refletir a assimetria atual. O valor no final do estágio reflete a assimetria do estágio final. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e observabilityGroup (job_performance) Estatísticas válidas: média, máximo, mínimo, porcentag em Unidade: Contagem	

Métrica	Descrição	Categoria
glue.driver.skewness.job	Categoria métrica: job_performance A assimetria do trabalho é a média ponderada da assimetria do estágio do trabalho. A média ponderada dá mais peso aos estágios que demoram mais para serem executados. Isso é para evitar situações extremas quando um estágio muito assimétrico está realmente funcionando por um tempo muito curto em relação a outros estágios (e, portanto, sua assimetria não é significativa para o desempenho geral do trabalho e não vale a pena tentar resolver sua assimetria). Essa métrica é atualizada após a conclusão de cada estágio e, portanto, o último valor reflete a assimetria geral real do trabalho. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e observabilityGroup (job_performance)	job_performance

Métrica	Descrição	Categoria
	Estatísticas válidas: média, máximo, mínimo, porcentag em Unidade: Contagem	
glue.succeed.ALL	Categoria métrica: erro Número total de execuções de trabalhos com êxito para completar o quadro das categorias de falhas Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (count) e ObservabilityGroup (error) Estatística válida: SUM Unidade: Contagem	erro

Métrica	Descrição	Categoria
glue.error.ALL	Categoria métrica: erro Número total de erros de execução de trabalho para completar o quadro das categorias de falhas Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (count) e ObservabilityGroup (error) Estatística válida: SUM Unidade: Contagem	erro

Métrica	Descrição	Categoria
glue.error.[error category]	Categoria métrica: erro Na verdade, este é um conjunto de métricas que são atualizadas somente quando a execução de um trabalho falha. A categoriz ação de erros ajuda na triagem e na depuração . Quando a execução de uma tarefa falha, o erro que causa a falha é categoriz ado e a métrica da categoria de erro correspondente é definida como 1. Isso ajuda a realizar análises de falhas ao longo do tempo, bem como a análise geral de erros de trabalhos para identific ar as categorias de falhas mais comuns e começar a resolvê-las. O AWS Glue tem 28 categorias de erro, incluindo as categorias de erro OUT_OF_MEMORY (driver e executor), PERMISSION, SYNTAX e THROTTLING. As categorias de erro também incluem as categorias de erro COMPILATION, LAUNCH e TIMEOUT. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de	erro

Métrica	Descrição	Categoria
	JobRun. ou ALL), Type (count) e ObservabilityGroup (error) Estatística válida: SUM Unidade: Contagem	
glue.driver.workerUtilization	Categoria métrica: resource_utilization A porcentagem dos trabalhadores alocados que são realmente usados. Se não for bom, o ajuste de escala automático pode ajudar. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatísticas válidas: média, máximo, mínimo, porcentagem Unidade: porcentagem	resource_utilization

Métrica	Descrição	Categoria
glue.driver.memory.heap.[available | used]	Categoria métrica: resource_utilization A memória heap disponível/usado do driver durante a execução do trabalho. Isso ajuda a entender as tendências de uso da memória, especialmente ao longo do tempo, o que pode ajudar a evitar possíveis falhas, além de depurar falhas relacionadas à memória. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: bytes	resource_utilization

Métrica	Descrição	Categoria
glue.driver.memory.heap.used.percentage	Categoria métrica: resource_utilization A memória heap usada pelo driver (%) durante a execução do trabalho. Isso ajuda a entender as tendências de uso da memória, especialmente ao longo do tempo, o que pode ajudar a evitar possíveis falhas, além de depurar falhas relacionadas à memória. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: porcentagem	resource_utilization

Métrica	Descrição	Categoria
glue.driver.memory.non-heap.[available | used]	Categoria métrica: resource_utilization A memória não heap disponível/usado do driver durante a execução do trabalho. Isso ajuda a entender as tendências de uso da memória, especialm ente ao longo do tempo, o que pode ajudar a evitar possíveis falhas, além de depurar falhas relacionadas à memória. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: bytes	resource_utilization

Métrica	Descrição	Categoria
glue.driver.memory.non-heap .used.percentage	Categoria métrica: resource_ utilization A memória não heap usada pelo driver (%) durante a execução do trabalho. Isso ajuda a entender as tendência s de uso da memória, especialmente ao longo do tempo, o que pode ajudar a evitar possíveis falhas, além de depurar falhas relaciona das à memória. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: porcentagem	resource_utilization

Métrica	Descrição	Categoria
glue.driver.memory.total.[available | used]	Categoria métrica: resource_utilization A memória total do driver disponível/usada durante a execução do trabalho. Isso ajuda a entender as tendências de uso da memória, especialmente ao longo do tempo, o que pode ajudar a evitar possíveis falhas, além de depurar falhas relacionadas à memória. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: bytes	resource_utilization

Métrica	Descrição	Categoria
glue.driver.memory.total.used.percentage	Categoria métrica: resource_utilization A memória total usada pelo driver (%) durante a execução do trabalho. Isso ajuda a entender as tendências de uso da memória, especialmente ao longo do tempo, o que pode ajudar a evitar possíveis falhas, além de depurar falhas relacionadas à memória. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: porcentagem	resource_utilization

Métrica	Descrição	Categoria
glue.ALL.memory.heap.[available | used]	Categoria métrica: resource_utilization A memória heap disponível/usada dos executores. ALL significa todos os executores. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: bytes	resource_utilization
glue.ALL.memory.heap.used.percentage	Categoria métrica: resource_utilization A memória heap usada dos executores (%). ALL significa todos os executores. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: porcentagem	resource_utilization

Métrica	Descrição	Categoria
glue.ALL.memory.non-heap.[available | used]	Categoria métrica: resource_utilization A memória não heap disponível/usada dos executores. ALL significa todos os executores. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: bytes	resource_utilization
glue.ALL.memory.non-heap.used.percentage	Categoria métrica: resource_utilization A memória não heap usada (%) dos executores. ALL significa todos os executores. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: porcentagem	resource_utilization

Métrica	Descrição	Categoria
glue.ALL.memory.total.[available | used]	Categoria métrica: resource_utilization A memória total disponível/usada dos executores. ALL significa todos os executores. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: bytes	resource_utilization
glue.ALL.memory.total.used.percentage	Categoria métrica: resource_utilization A memória total usada (%) dos executores. ALL significa todos os executores. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: porcentagem	resource_utilization

Métrica	Descrição	Categoria
glue.driver.disk.[available_GB | used_GB]	Categoria métrica: resource_utilization O espaço em disco disponível/usado do driver durante a execução do trabalho. Isso ajuda a entender as tendências de uso do disco, especialmente ao longo do tempo, o que pode ajudar a evitar possíveis falhas, além de depurar falhas relacionadas à falta de espaço em disco. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: gigabytes	resource_utilization

Métrica	Descrição	Categoria
glue.driver.disk.used.percentage]	Categoria métrica: resource_utilization O espaço em disco disponível/usado do driver durante a execução do trabalho. Isso ajuda a entender as tendências de uso do disco, especialmente ao longo do tempo, o que pode ajudar a evitar possíveis falhas, além de depurar falhas relacionadas à falta de espaço em disco. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: porcentagem	resource_utilization

Métrica	Descrição	Categoria
glue.ALL.disk.[available_GB | used_GB]	Categoria métrica: resource_utilization O espaço em disco disponível/usado dos executores. ALL significa todos os executores. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: gigabytes	resource_utilization
glue.ALL.disk.used.percentage	Categoria métrica: resource_utilization O espaço em disco disponível/usado/usado(%) dos executores. ALL significa todos os executores. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Estatística válida: média Unidade: porcentagem	resource_utilization

Métrica	Descrição	Categoria
glue.driver.bytesRead	Categoria métrica: throughput O número de bytes lidos por fonte de entrada nessa execução de trabalho, bem como para TODAS as fontes. Isso ajuda a entender o volume de dados e suas mudanças ao longo do tempo, o que ajuda a resolver problemas como assimetria de dados. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge), ObservabilityGroup (resource_utilization) e Source (local da fonte de dados) Estatística válida: média Unidade: bytes	throughput

Métrica	Descrição	Categoria
glue.driver.[recordsRead | filesRead]	Categoria métrica: throughput O número de registros/ arquivos lidos por fonte de entrada nessa execução de trabalho, bem como para TODAS as fontes. Isso ajuda a entender o volume de dados e suas mudanças ao longo do tempo, o que ajuda a resolver problemas como assimetria de dados. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge), ObservabilityGroup (resource_utilization) e Source (local da fonte de dados) Estatística válida: média Unidade: Contagem	throughput

Métrica	Descrição	Categoria
glue.driver.partitionsRead	Categoria métrica: throughput O número de partições lidas por fonte de entrada do Amazon S3 nessa execução de trabalho, bem como para TODAS as fontes. Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge), ObservabilityGroup (resource_utilization) e Source (local da fonte de dados) Estatística válida: média Unidade: Contagem	throughput

Métrica	Descrição	Categoria
glue.driver.bytesWrittten	Categoria métrica: throughput O número de bytes gravados por coletor de saída nessa execução de trabalho, bem como para TODOS os coletores. Isso ajuda a compreender o volume de dados e como ele evolui ao longo do tempo, o que ajuda a resolver problemas como a assimetria do processamento Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge), ObservabilityGroup (resource_utilization) e Source (local do coletor de dados) Estatística válida: média Unidade: bytes	throughput

Métrica	Descrição	Categoria
glue.driver.[recordsWritten | filesWritten]	Categoria métrica: throughput O número de registros/ arquivos gravados por coletor de saída nessa execução de trabalho, bem como para TODOS os coletores . Isso ajuda a compreend er o volume de dados e como ele evolui ao longo do tempo, o que ajuda a resolver problemas como a assimetria do processamento Dimensões válidas: JobName (o nome do trabalho do AWS Glue), JobRunId (a ID de JobRun. ou ALL), Type (gauge), ObservabilityGroup (resource_utilization) e Source (local do coletor de dados) Estatística válida: média Unidade: Contagem	throughput

Categorias de erros

Categorias de erros	Descrição
COMPILATION_ERROR	Os erros surgem durante a compilação do código Scala.

Categorias de erros	Descrição
CONNECTION_ERROR	Os erros surgem durante a conexão com um serviço/host remoto/serviço de banco de dados etc.
DISK_NO_SPACE_ERROR	Os erros surgem quando não há mais espaço em disco no driver/executor.
OUT_OF_MEMORY_ERROR	Os erros surgem quando não há mais espaço na memória do driver/executor.
IMPORT_ERROR	Os erros surgem ao importar dependências.
INVALID_ARGUMENT_ERROR	Erros surgem quando os argumentos de entrada são inválidos/ilegais.
PERMISSION_ERROR	Os erros surgem quando não há permissão para serviços, dados etc.
RESOURCE_NOT_FOUND_ERROR	Os erros surgem quando os dados, a localizaç ão etc. não existem.
QUERY_ERROR	Os erros surgem da execução da consulta do Spark SQL.
SYNTAX_ERROR	Os erros surgem quando há um erro de sintaxe no script.
THROTTLING_ERROR	Os erros surgem ao atingir a limitação de simultaneidade de serviços ou ao exceder a limitação de cotas de serviço.
DATA_LAKE_FRAMEWORK_ERROR	Os erros surgem de estruturas de data lake com suporte nativo do AWS Glue, como Hudi, Iceberg etc.
UNSUPPORTED_OPERATION_ERROR	Os erros surgem ao realizar uma operação não compatível.

Categorias de erros	Descrição
RESOURCES_ALREADY_EXISTS_ERROR	Os erros surgem quando um recurso a ser criado ou adicionado já existe.
GLUE_INTERNAL_SERVICE_ERROR	Os erros surgem quando há um problema de serviço interno do AWS Glue.
GLUE_OPERATION_TIMEOUT_ERROR	Os erros surgem quando uma operação do AWS Glue atinge o tempo limite.
GLUE_VALIDATION_ERROR	Os erros surgem quando um valor exigido não pôde ser validado para um trabalho do AWS Glue.
GLUE_JOB_BOOKMARK_VERSION_MISMATCH_ERROR	Os erros surgem quando o mesmo trabalho transcreve o mesmo bucket de origem e grava no mesmo destino ou em outro simultaneamente (simultaneidade >1)
LAUNCH_ERROR	Os erros surgem durante a fase de lançamento do trabalho do AWS Glue.
DYNAMODB_ERROR	Os erros genéricos surgem do serviço do Amazon DynamoDB.
GLUE_ERROR	Os erros genéricos surgem do serviço do AWS Glue.
LAKEFORMATION_ERROR	Os erros genéricos surgem do serviço do AWS Lake Formation.
REDSHIFT_ERROR	Os erros genéricos surgem do serviço do Amazon Redshift.
S3_ERROR	Os erros genéricos surgem do serviço do Amazon S3.
SYSTEM_EXIT_ERROR	Erro genérico de saída do sistema.

Categorias de erros	Descrição
TIMEOUT_ERROR	Os erros genéricos surgem quando o trabalho falha devido ao tempo limite da operação.
UNCLASSIFIED_SPARK_ERROR	Os erros genéricos surgem do Spark.
UNCLASSIFIED_ERROR	Categoria de erro padrão.

Limitações

> **Note**
>
> O `glueContext` deve ser inicializado para publicar as métricas.

Na Dimensão de origem, o valor é o caminho ou o nome da tabela do Amazon S3, dependendo do tipo de fonte. Além disso, se a origem for JDBC e a opção de consulta for usada, a string de consulta será definida na dimensão de origem. Se o valor tiver mais de 500 caracteres, ele será reduzido em 500 caracteres. Abaixo, estão as limitações do valor:

- Os caracteres não ASCII serão removidos.
- Se o nome da fonte não contiver nenhum caractere ASCII, ele será convertido para <non-ASCII input>.

Limitações e considerações sobre métricas de throughput

- O DataFrame e o DynamicFrame baseado em DataFrame (por exemplo, JDBC, leitura de parquet no Amazon S3) são compatíveis, no entanto, o DynamicFrame baseado em RDD (por exemplo, leitura de csv, json no Amazon S3 etc.) não é compatível. Tecnicamente, todas as leituras e gravações visíveis na UI do Spark são compatíveis.
- A métrica `recordsRead` será emitida se a fonte de dados for uma tabela de catálogo e o formato for JSON, CSV, text ou Iceberg.
- As métricas `glue.driver.throughput.recordsWritten`, `glue.driver.throughput.bytesWritten` e `glue.driver.throughput.filesWritten` não estão disponíveis nas tabelas JDBC e Iceberg.

- As métricas podem estar atrasadas. Se o trabalho for concluído em cerca de um minuto, talvez não haja métricas de throughput nas métricas do Amazon CloudWatch.

Monitoramento e depuração de trabalho

Você pode coletar métricas sobre trabalhos do AWS Glue e visualizá-las nos consoles do AWS Glue e do Amazon CloudWatch para identificar e corrigir problemas. A criação do perfil dos seus trabalhos do AWS Glue requer as seguintes etapas:

1. Habilitar métricas:

   a. Habilite a opção Job metrics (Métricas de trabalho) na definição do trabalho. Você pode habilitar a criação de perfil no console do AWS Glue ou como um parâmetro para o trabalho. Para obter mais informações, consulte Definir propriedades de trabalho para trabalhos do Spark ou Parâmetros de trabalho do AWS Glue.

   b. Habilite a opção Métricas de observabilidade do AWS Glue na definição do trabalho. É possível habilitar a observabilidade no console do AWS Glue ou como parâmetro para o trabalho. Para ter mais informações, consulte Monitoramento com métricas de observabilidade do AWS Glue.

2. Confirme se o script do trabalho inicializa um `GlueContext`. Por exemplo, o snippet de script a seguir inicializa um `GlueContext` e mostra onde o código perfilado é colocado no script. Este formato geral é usado nos cenários de depuração a seguir.

```
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. Execute o trabalho.
4. Visualize as métricas:
   a. Visualize as métricas de trabalho no console do AWS Glue e identifique métricas anormais para o driver ou um executor.
   b. Verifique as métricas de observabilidade na página de monitoramento da execução de trabalhos, na página de detalhes da execução de trabalhos ou no Amazon CloudWatch. Para ter mais informações, consulte Monitoramento com métricas de observabilidade do AWS Glue.
5. Refine a causa raiz usando a métrica identificada.
6. Opcionalmente, confirme a causa raiz usando o fluxo de log do driver identificado ou executor do trabalho.

Casos de uso para métricas de observabilidade do AWS Glue

- Depurar exceções OOM e anomalias de trabalho
- Depurar estágios exigentes e tarefas de retardatário
- Monitorar o progresso de vários trabalhos
- Monitorar planejamento de capacidade de DPU
- Como usar a observabilidade do AWS Glue para monitorar a utilização de recursos para reduzir custos

Depurar exceções OOM e anomalias de trabalho

Você pode depurar exceções de memória insuficiente (OOM) e anomalias de trabalho no AWS Glue. As seções a seguir descrevem cenários para depuração de exceções de memória insuficiente do driver do Apache Spark ou um executor do Spark.

- Depurar uma exceção OOM do driver
- Depurar uma exceção OOM do executor

Depurar uma exceção OOM do driver

Nesse cenário, um trabalho do Spark está lendo um grande número de arquivos pequenos do Amazon Simple Storage Service (Amazon S3). Ele converte os arquivos no formato Apache Parquet e, em seguida, os grava no Amazon S3. O driver do Spark está sendo executado sem memória. A entrada de dados do Amazon S3 tem mais de 1 milhão de arquivos em diferentes partições do Amazon S3.

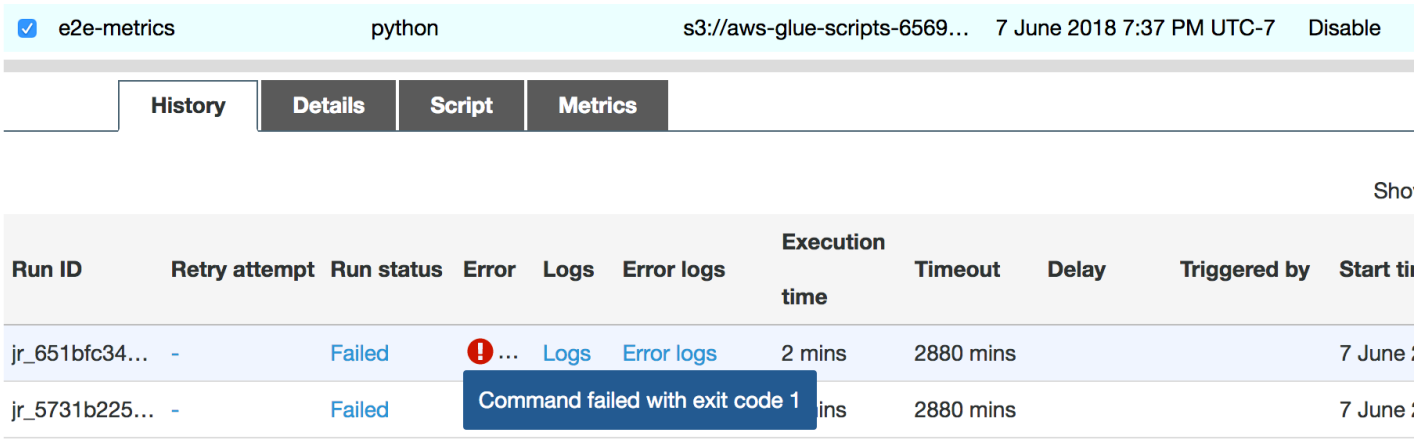
O código perfilado é assim:

```
data = spark.read.format("json").option("inferSchema", False).load("s3://input_path")
data.write.format("parquet").save(output_path)
```

Visualizar as métricas perfiladas no console do AWS Glue

O gráfico a seguir mostra o uso de memória como uma porcentagem para o driver e executores. Esse uso é representado como um ponto de dados cuja média é calculada com base nos valores relatados no último minuto. Você pode ver no perfil de memória do trabalho que a memória do driver ultrapassa o limite de 50% de uso rapidamente. Por outro lado, a média de uso de memória em todos os executores ainda é menos de 4%. Isso mostra claramente anomalia com a execução do driver nesse trabalho do Spark.

A execução do trabalho logo falha, e o seguinte erro é exibido na guia History (Histórico) no console do AWS Glue: Command Failed with Exit Code 1 (Falha do comando com código de saída 1). Essa string de erro significa que o trabalho falhou devido a um erro sistêmico, que neste caso é a falta de memória do driver.



No console, escolha o link Error logs (Logs de erro) na guia History (Histórico) para confirmar a descoberta do CloudWatch Logs sobre a OOM do driver. Procure "**`Error`**" nos logs de erro do trabalho para confirmar que foi uma exceção OOM que falhou o trabalho:

```
# java.lang.OutOfMemoryError: Java heap space
# -XX:OnOutOfMemoryError="kill -9 %p"
```

```
# Executing /bin/sh -c "kill -9 12039"...
```

Na guia History (Histórico) do trabalho, escolha Logs. Você pode encontrar as seguintes evidências da execução do driver no CloudWatch Logs no início do trabalho. O driver do Spark tenta listar todos os arquivos em todos os diretórios, cria um `InMemoryFileIndex` e inicia uma tarefa por arquivo. Isso, por sua vez, resulta no driver do Spark precisar manter uma grande quantidade de estado na memória para rastrear todas as tarefas. Ele armazena em cache a lista completa de um grande número de arquivos para o índice na memória, o que resulta em um driver OOM.

Corrigir o processamento de vários arquivos usando agrupamento

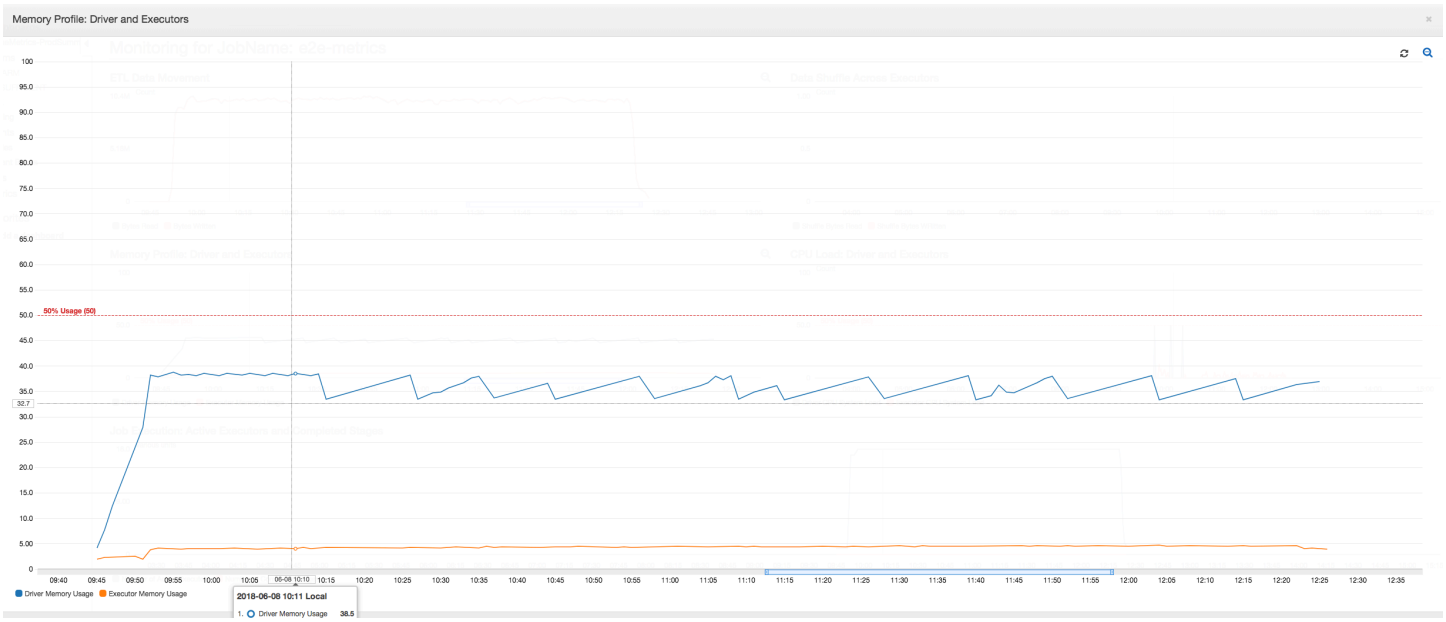
Você pode corrigir o processamento de vários arquivos usando o recurso de agrupamento no AWS Glue. O agrupamento é habilitado automaticamente quando você usa quadros dinâmicos e quando o conjunto de dados de entrada tem um grande número de arquivos (mais de 50.000). Agrupar permite reunir vários arquivos em um grupo, e isso permite que uma tarefa processe todo o grupo em vez de um único arquivo. Como resultado, o driver do Spark armazena significativamente menos estado na memória para rastrear menos tarefas. Para obter mais informações sobre como habilitar manualmente o agrupamento para o seu conjunto de dados, consulte Ler arquivos de entrada em grupos maiores.

Para verificar o perfil de memória do trabalho do AWS Glue, crie o perfil do código a seguir com agrupamento habilitado:

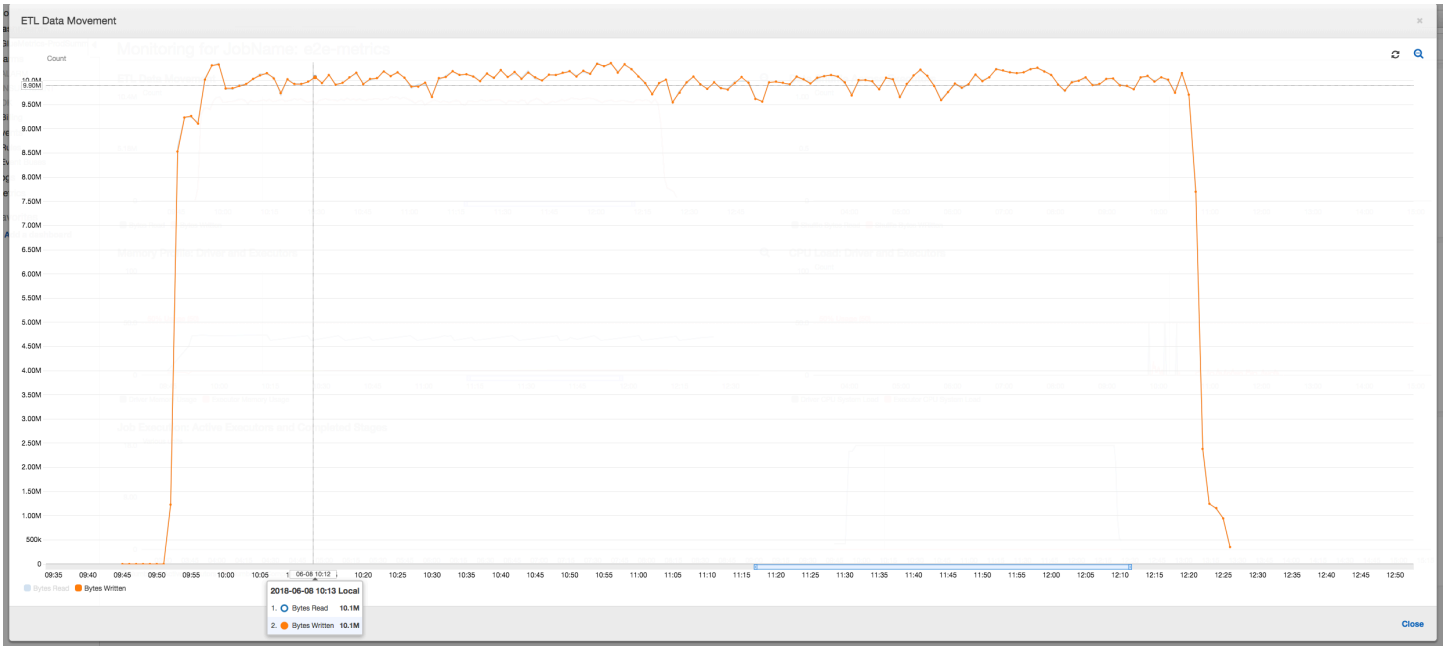
```
df = glueContext.create_dynamic_frame_from_options("s3", {'paths': ["s3://input_path"],
 "recurse":True, 'groupFiles': 'inPartition'}, format="json")
datasink = glueContext.write_dynamic_frame.from_options(frame = df, connection_type
 = "s3", connection_options = {"path": output_path}, format = "parquet",
 transformation_ctx = "datasink")
```

Você pode monitorar o perfil de memória e a movimentação de dados ETL no perfil de trabalho do AWS Glue.

O driver é executado abaixo do limite de 50% de uso de memória ao longo de toda a duração do trabalho do AWS Glue. Os executores transmitem os dados do Amazon S3, os processam e os gravam no Amazon S3. Como resultado, eles consomem menos de 5% de memória a qualquer momento.



O perfil de movimentação de dados abaixo mostra o número total de bytes do Amazon S3 que são lidos e gravados no último minuto por todos os executores conforme o trabalho progride. Ambos seguem um padrão semelhante conforme os dados são transmitidos em todos os executores. O trabalho termina o processamento de um milhão de arquivos em menos de três horas.



Depurar uma exceção OOM do executor

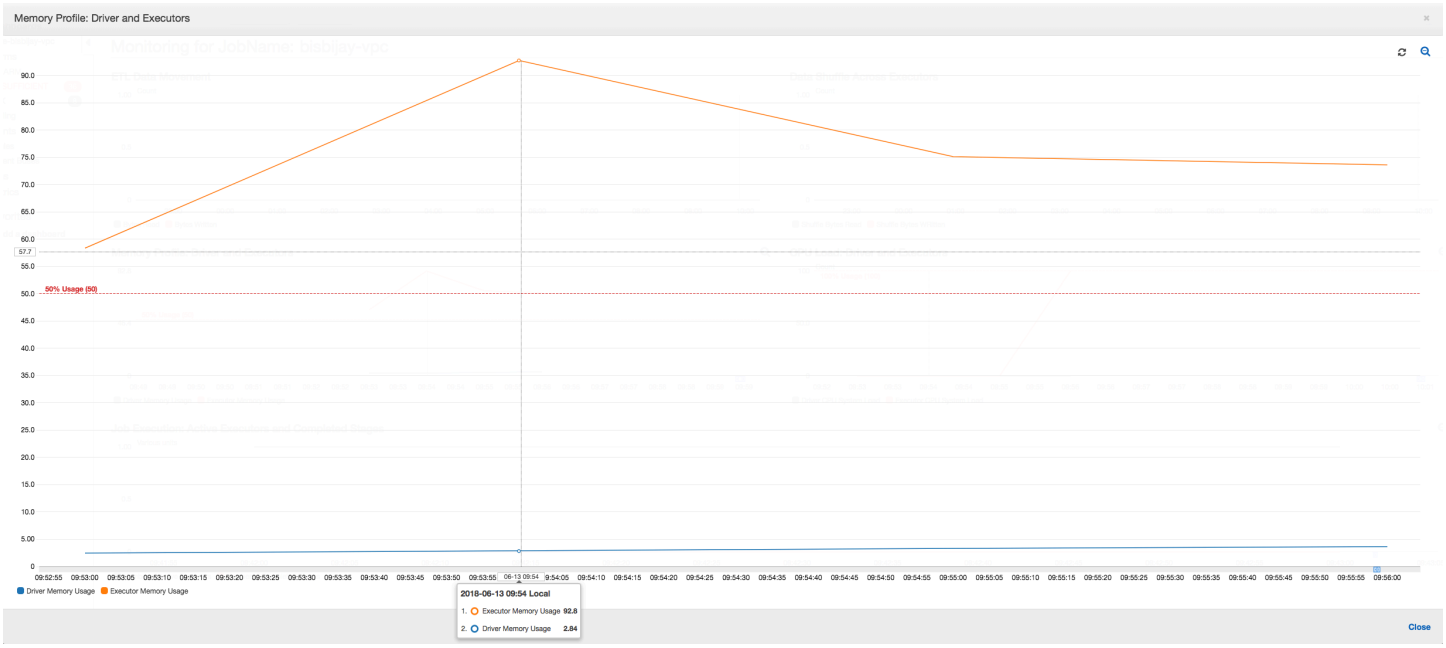
Nesse cenário, você pode saber como depurar exceções OOM que podem ocorrer em executores do Apache Spark. O código a seguir usa o leitor Spark MySQL para ler uma grande tabela de cerca de 34 milhões de linhas em um dataframe do Spark. Em seguida, ele grava no Amazon S3 em formato

Parquet. Você pode fornecer as propriedades da conexão e usar as configurações padrão do Spark para ler a tabela.

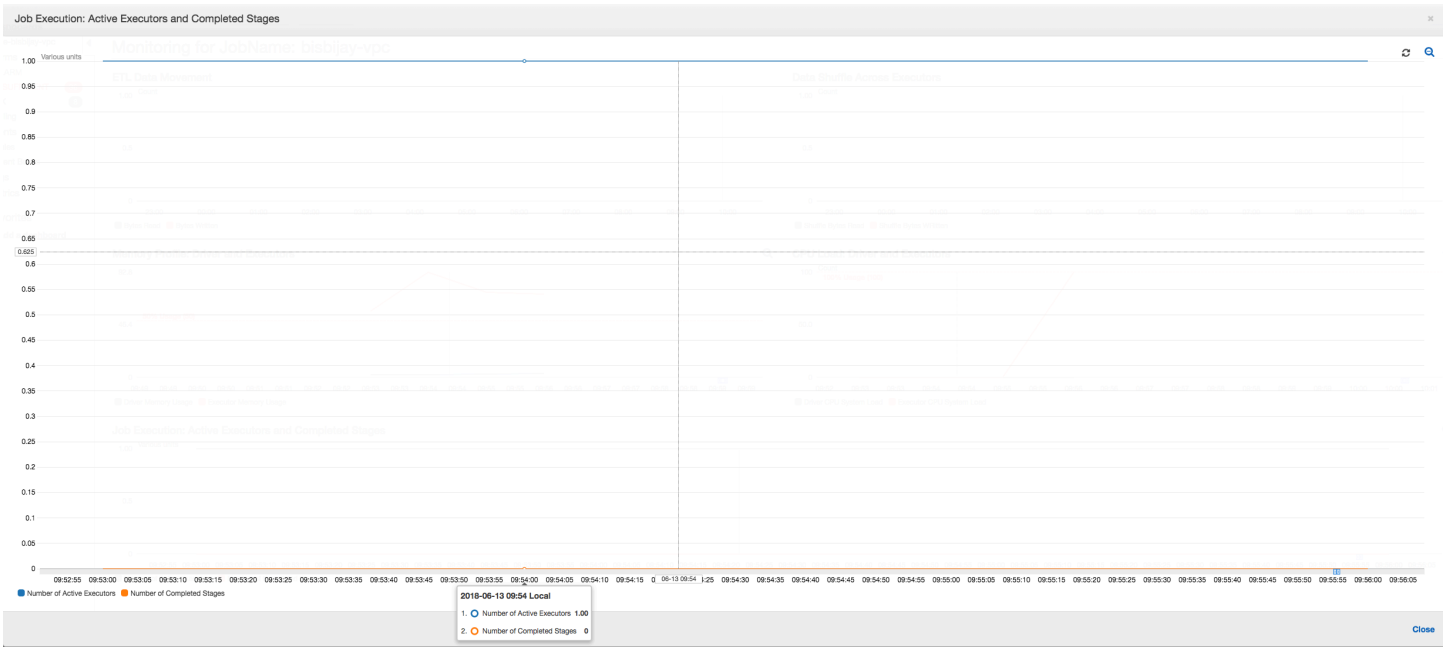
```
val connectionProperties = new Properties()
connectionProperties.put("user", user)
connectionProperties.put("password", password)
connectionProperties.put("Driver", "com.mysql.jdbc.Driver")
val sparkSession = glueContext.sparkSession
val dfSpark = sparkSession.read.jdbc(url, tableName, connectionProperties)
dfSpark.write.format("parquet").save(output_path)
```

Visualizar as métricas perfiladas no console do AWS Glue

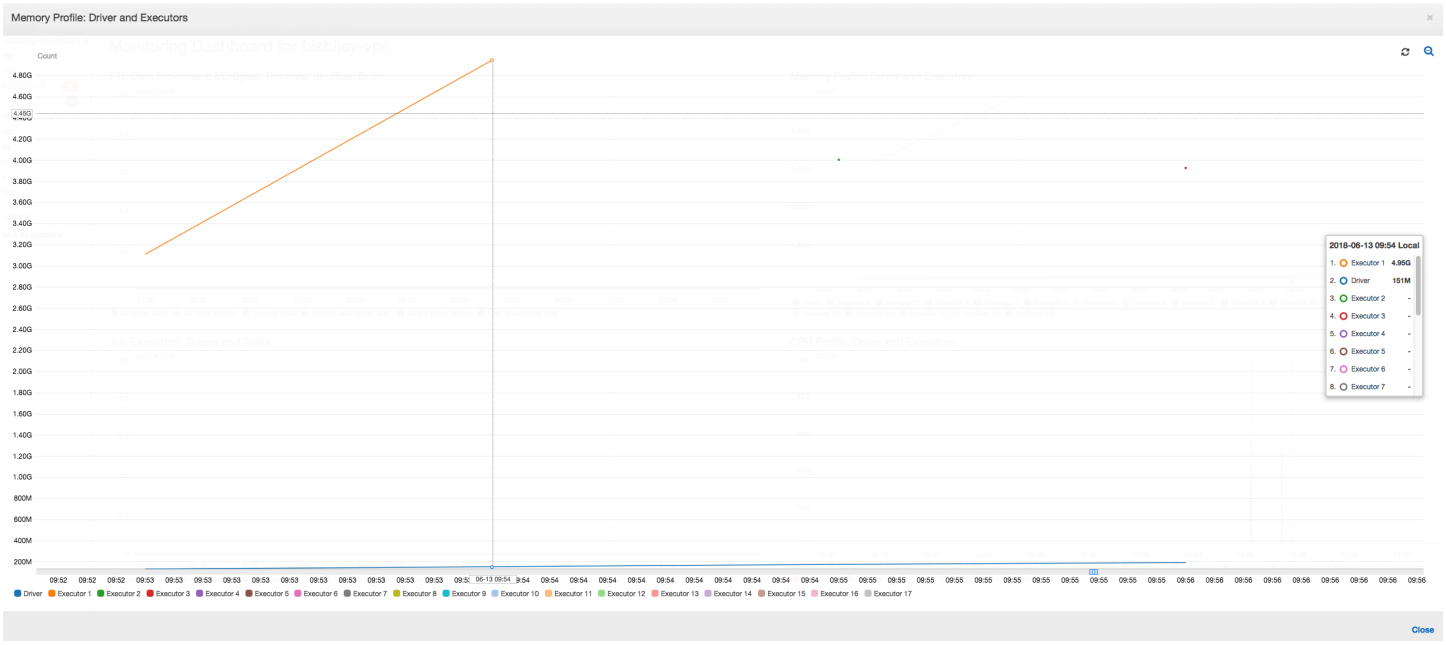
Se a inclinação do gráfico de uso de memória for positiva e ultrapassar 50%, e se ocorrer uma falha no trabalho antes que a próxima métrica seja emitida, o esgotamento da memória será uma causa provável. O seguinte gráfico mostra que, dentro de um minuto da execução, a média de uso de memória em todos os executores ultrapassa rapidamente 50%. O uso atinge até 92%, e o contêiner que executa o executor é interrompido pelo Apache Hadoop YARN.



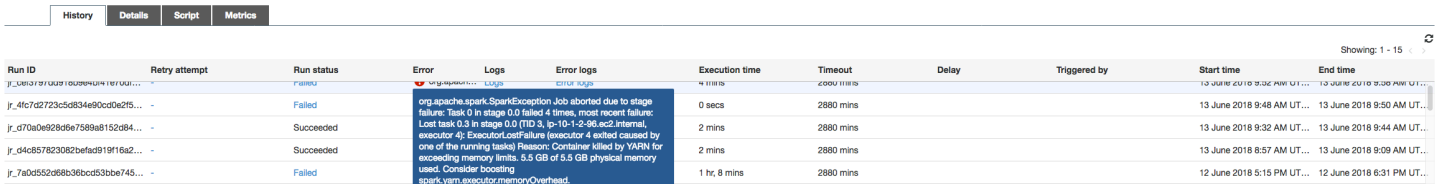
Como o seguinte gráfico mostra, há sempre um único executor em execução até o trabalho falhar. Isso ocorre porque um novo executor é iniciado para substituir o executor eliminado. As leituras da fonte de dados JDBC não são paralelizadas por padrão porque isso exigiria particionamento de tabela em uma coluna e abrir várias conexões. Como resultado, apenas um executor lê na tabela completa sequencialmente.



Como o seguinte gráfico mostra, o Spark tenta iniciar uma nova tarefa quatro vezes antes da falha no trabalho. Você pode ver o perfil de memória de três executores. Cada executor usa rapidamente toda sua memória. O quarto executor fica sem memória e o trabalho falha. Como resultado, a métrica não é relatada imediatamente.



Você pode confirmar na string de erro no console do AWS Glue que o trabalho falhou devido a exceções OOM, conforme mostrado na imagem a seguir.



Logs de saída de trabalho: para confirmação adicional da descoberta de uma exceção OOM de executor, consulte o CloudWatch Logs. Quando você pesquisar por **`Error`**, encontrará os quatro executores interrompidos aproximadamente na mesma janela de tempo, como mostrado no painel de métricas. Todos são encerrados pelo YARN à medida que excedem os limites de memória.

Executor 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.
```

Executor 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.
```

Executor 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.
```

Executor 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.
```

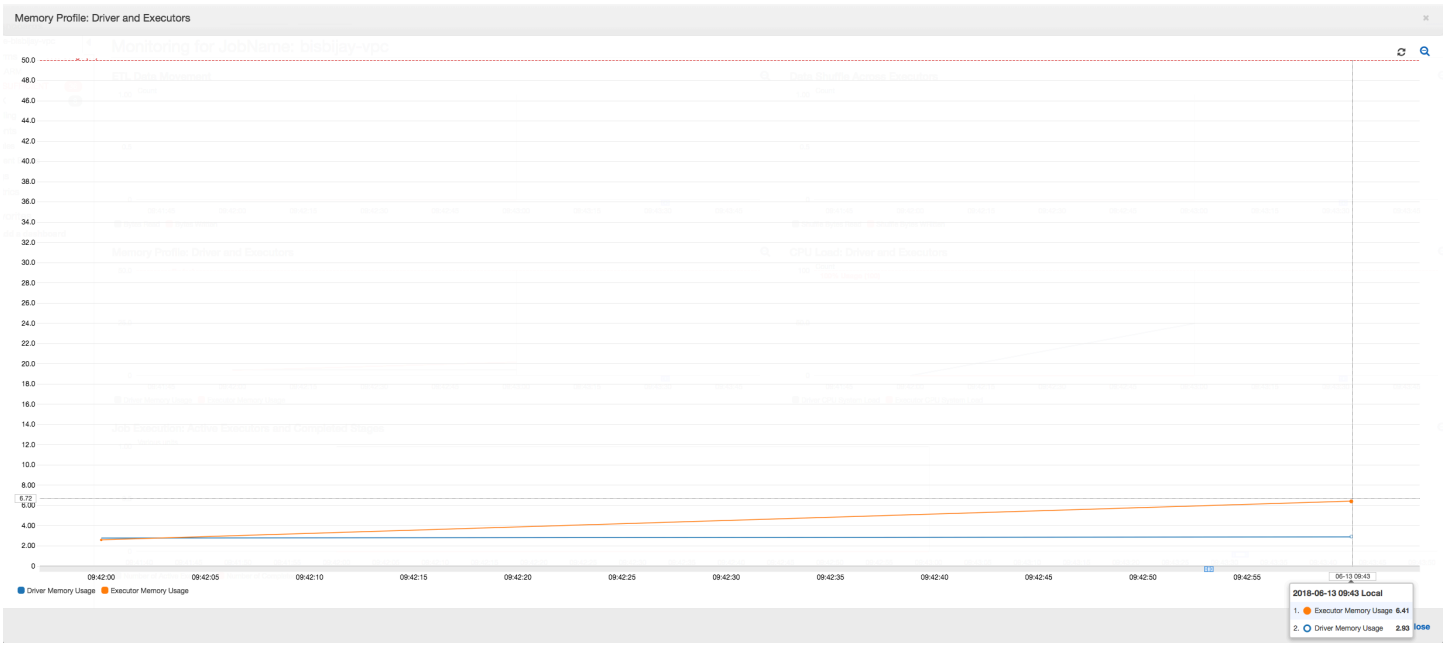
Corrigir a configuração do tamanho da busca usando quadros dinâmicos do AWS Glue

O executor ficou sem memória ao ler a tabela JDBC porque a configuração padrão para o tamanho da busca JDBC do Spark é zero. Isso significa que o driver JDBC no executor do Spark tenta buscar as 34 milhões de linhas no banco de dados em conjunto e as armazena em cache, embora o Spark transmita por meio de linhas, uma de cada vez. Com o Spark, você pode evitar esse cenário ao configurar o parâmetro de tamanho de busca como um valor padrão diferente de zero.

Você também pode corrigir esse problema usando os quadros dinâmicos do AWS Glue. Por padrão, os quadros dinâmicos usam um tamanho de busca de 1.000 linhas, que normalmente é um valor suficiente. Como resultado, o executor não consome mais de 7% da memória total. O trabalho do AWS Glue é concluído em menos de dois minutos com apenas um único executor. Embora usar quadros dinâmicos do AWS Glue seja a abordagem recomendada, também é possível definir o tamanho de busca usando a propriedade `fetchsize` do Apache Spark. Consulte o Guia do Spark SQL, DataFrames e conjuntos de dados.

```
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 perfiladas normais: a memória do executor com quadros dinâmicos do AWS Glue jamais excede o limite seguro, conforme mostrado na imagem a seguir. Ele transmite nas linhas do banco de dados e armazena em cache apenas 1.000 linhas no driver JDBC a qualquer momento. Não ocorre uma exceção de falta de memória.



Depurar estágios exigentes e tarefas de retardatário

Você pode usar perfis de trabalho do AWS Glue para identificar estágios exigentes e tarefas de retardatário em seus trabalhos de extração, transformação e carga (ETL). Uma tarefa de retardatário leva muito mais tempo do que o restante das tarefas em um estágio de um trabalho do AWS Glue. Como resultado, o estágio leva mais tempo para ser concluído, o que também atrasa o tempo total de execução do trabalho.

Reunir arquivos de entrada pequenos em arquivos de saída maiores

Uma tarefa de retardatário pode ocorrer quando há uma distribuição não uniforme de trabalho em diferentes tarefas, ou um desvio de dados resulta em uma tarefa processando mais dados.

Você pode criar o perfil do seguinte código (um padrão comum no Apache Spark) para reunir um grande número de arquivos pequenos em arquivos de saída maiores. Para este exemplo, o conjunto de dados de entrada é 32 GB de arquivos JSON com compactação Gzip. O conjunto de dados de saída tem aproximadamente 190 GB de arquivos JSON não compactados.

O código perfilado é assim:

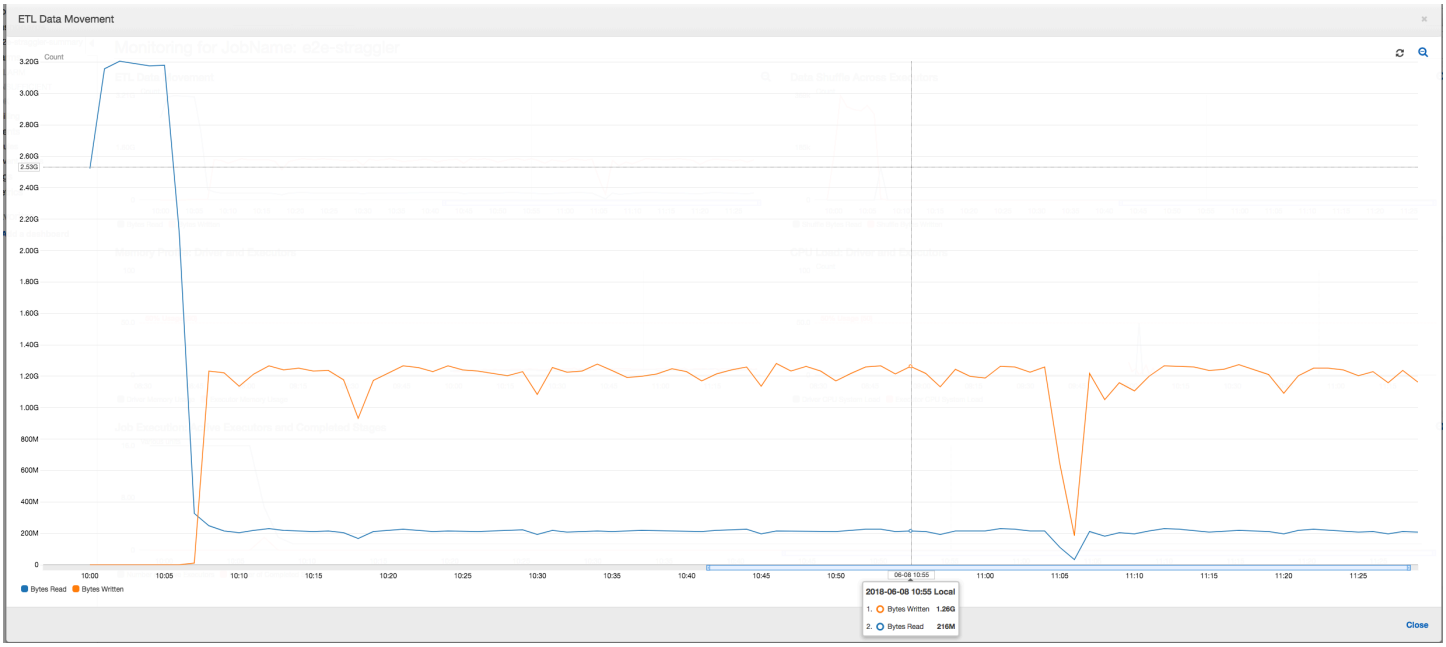
```
datasource0 = spark.read.format("json").load("s3://input_path")
df = datasource0.coalesce(1)
df.write.format("json").save(output_path)
```

Visualizar as métricas perfiladas no console do AWS Glue

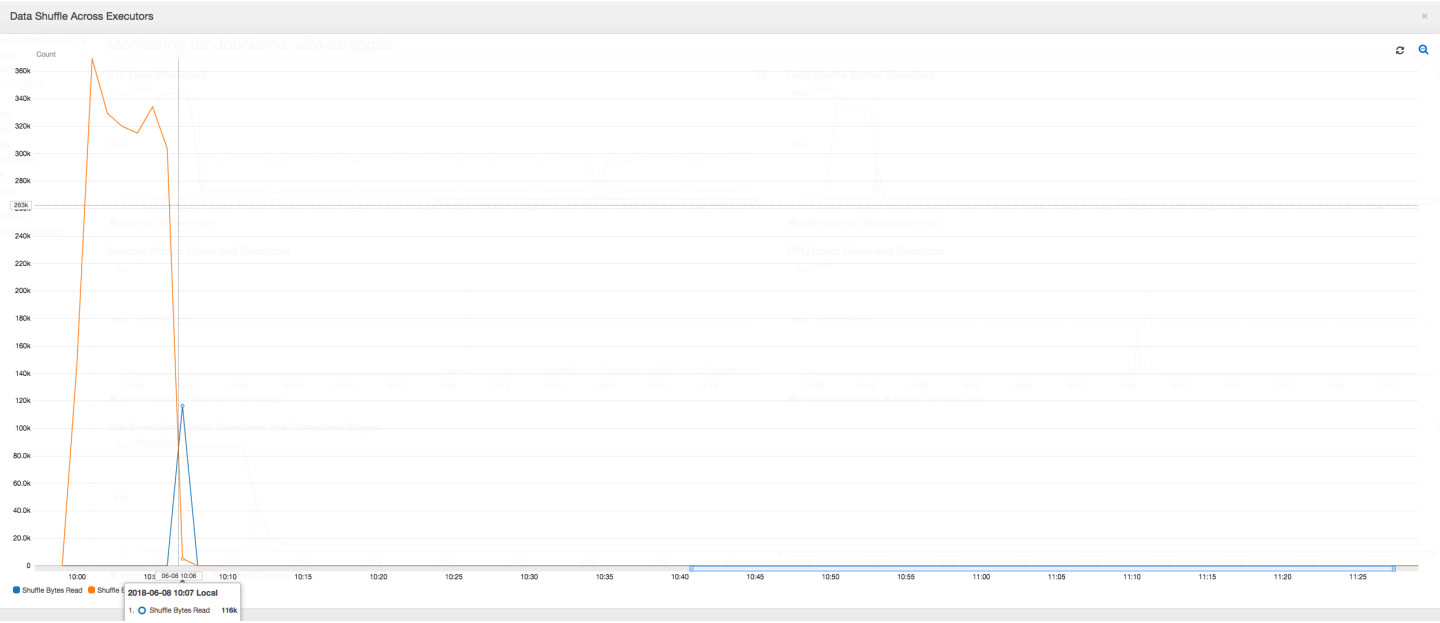
Você pode criar o perfil do trabalho para examinar quatro conjuntos de métricas diferentes:

- Movimentação de dados ETL
- Embaralhamento de dados em executores
- Execução de trabalho
- Perfil de memória

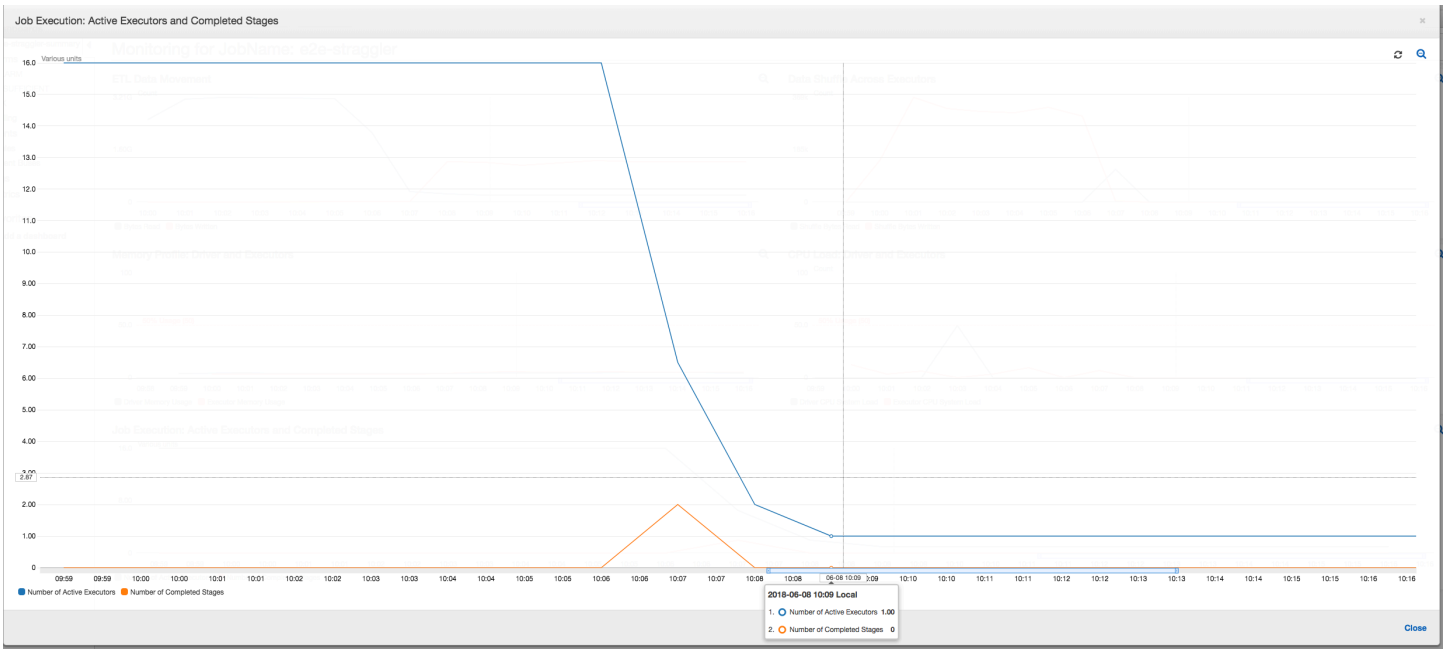
Movimentação de dados ETL: no perfil ETL Data Movement (Movimentação de dados ETL), os bytes são lidos bastante rapidamente por todos os executores no primeiro estágio concluído nos primeiros seis minutos. No entanto, o tempo total de execução do trabalho é de cerca de uma hora, principalmente consistindo em gravações de dados.



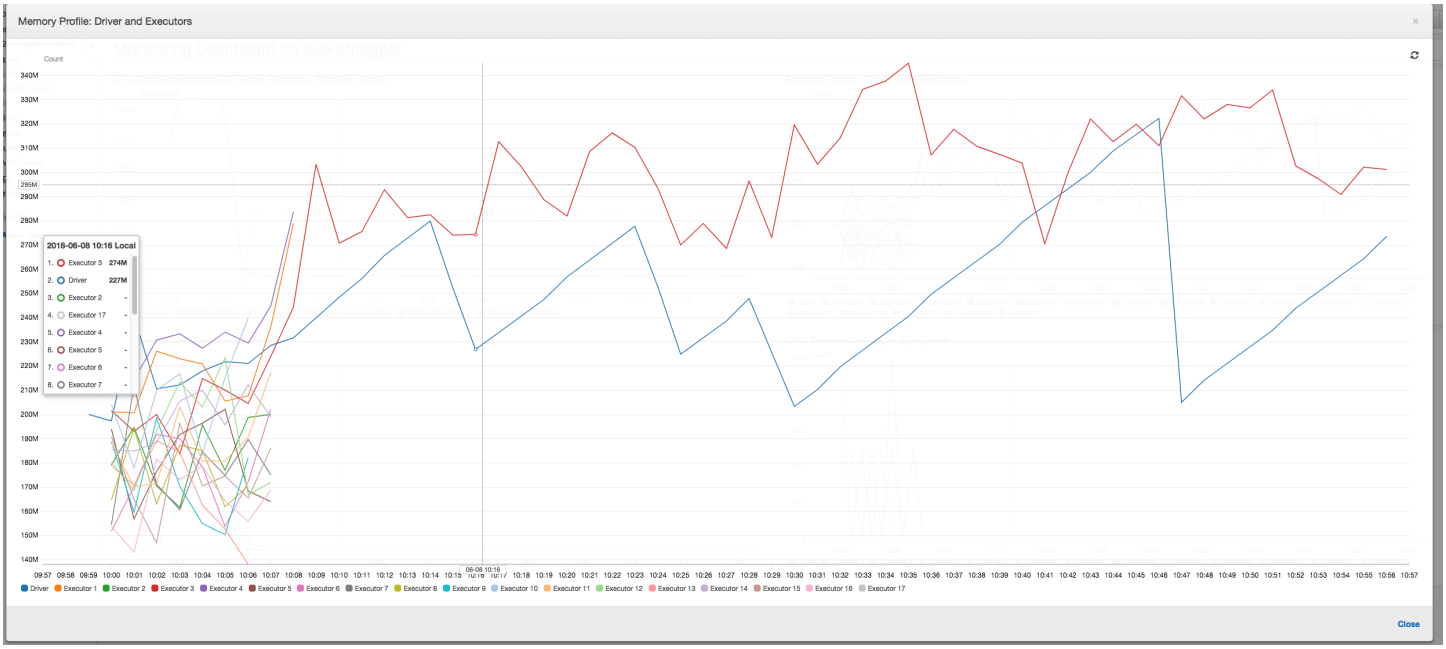
Embaralhamento de dados em executores: o número de bytes lidos e gravados durante o embaralhamento também mostra um pico antes da fase 2 terminar, conforme indicado pelas métricas Job Execution (Execução do trabalho) e Data Shuffle (Embaralhamento de dados). Depois que os dados são embaralhados em todos os executores, as leituras e gravações prosseguem no executor número 3 apenas.



Execução de trabalho: como mostrado no gráfico a seguir, todos os outros executores estão ociosos e acabam sendo desativados às 10h09. Nesse momento, o número total de executores diminui a apenas um. Isso claramente mostra que o executor número 3 consiste na tarefa de retardatário que está levando o tempo de execução mais longo e está contribuindo para a maior parte do tempo de execução do trabalho.



Perfil de memória: após os dois primeiros estágios, apenas o executor número 3 está ativamente consumindo memória para processar os dados. Os executores restantes estão simplesmente ociosos ou foram desativados logo após a conclusão dos dois primeiros estágios.



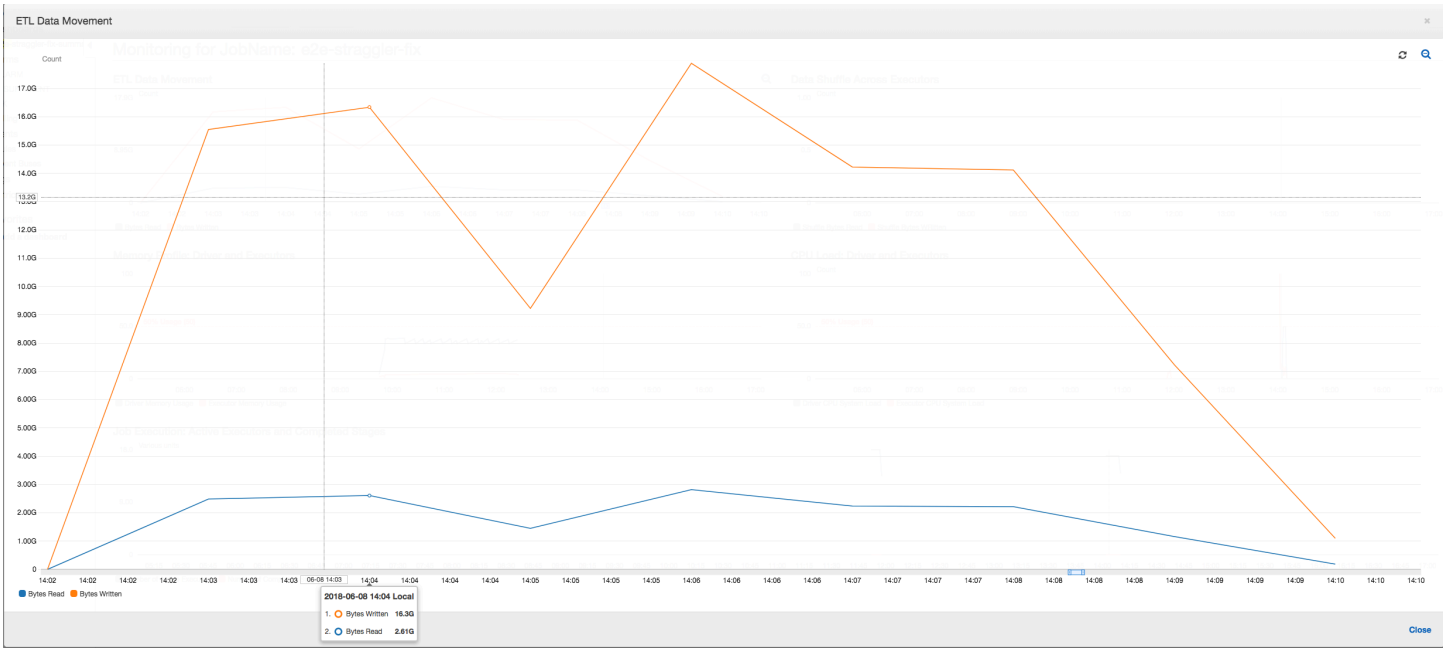
Corrigir executores retardatários usando agrupamento

Você pode evitar executores retardatários usando o recurso de agrupamento no AWS Glue. Use o agrupamento para distribuir os dados uniformemente em todos os executores e agrupar os arquivos em arquivos maiores usando todos os executores disponíveis no cluster. Para ter mais informações, consulte Ler arquivos de entrada em grupos maiores.

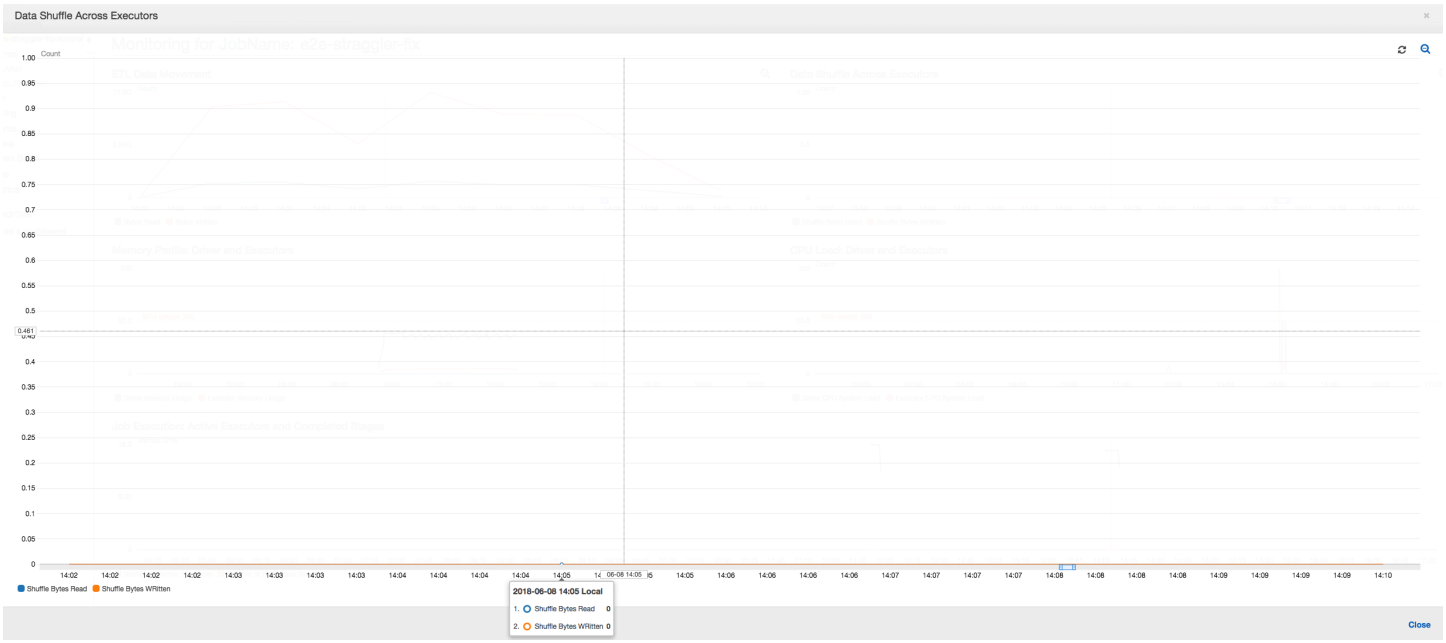
Para verificar as movimentações de dados ETL no AWS Glue, crie o perfil do código a seguir com agrupamento habilitado:

```
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")
```

Movimentação de dados ETL: as gravações de dados agora são transmitidas em paralelo com os dados lidos durante todo o tempo de execução do trabalho. Como resultado, o trabalho é concluído em oito minutos, muito mais rápido do que anteriormente.

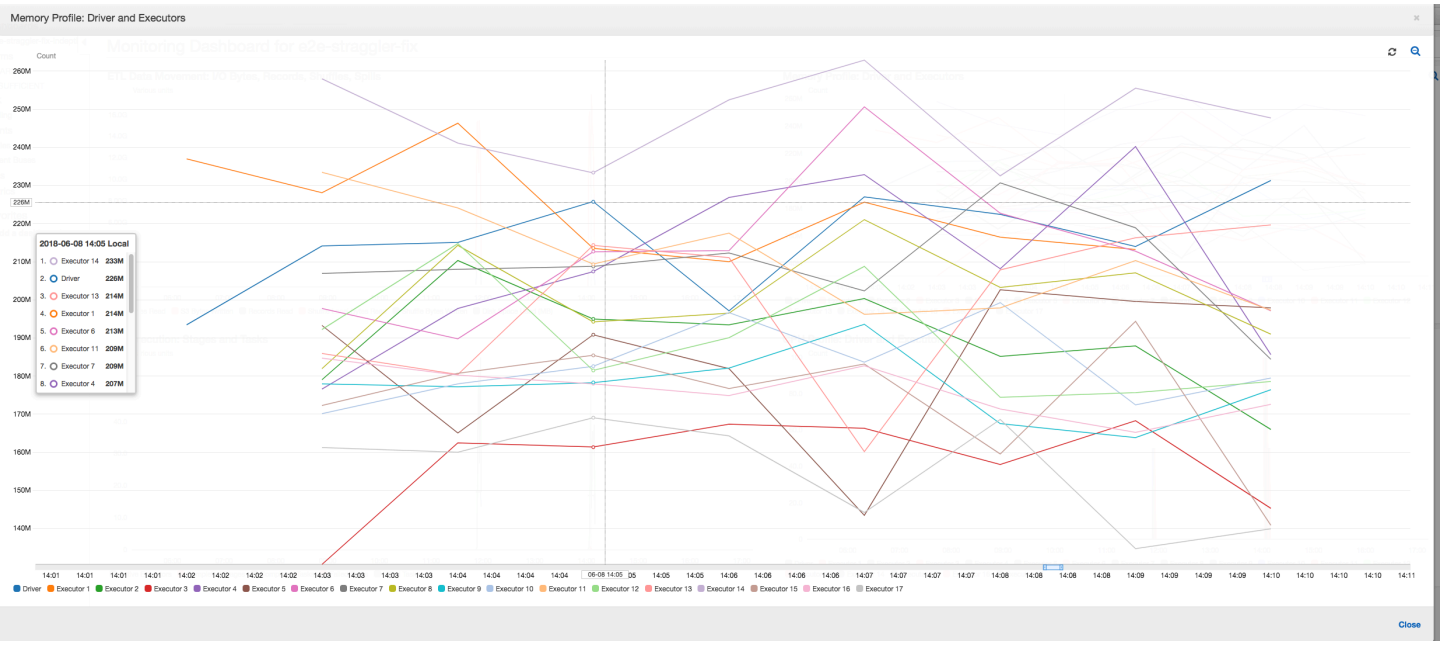


Embaralhamento de dados em executores: conforme os arquivos de entrada são agrupados durante as leituras usando o recurso de agrupamento, não há embaralhamento de dados dispendioso depois das leituras.



Execução do trabalho: as métricas de execução do trabalho mostram que o número total de executores ativos em execução e o processamento de dados permanece constante. Não há um único retardatário no trabalho. Todos os executores estão ativos e não são desativados até a conclusão do trabalho. Como não há embaralhamento intermediário dos dados nos executores, há somente uma única fase do trabalho.

Perfil de memória: as métricas mostram o consumo de memória ativa em todos os executores, reconfirmando que há atividade em todos eles. À medida que os dados são transmitidos e gravados em paralelo, o total de espaço de memória de todos os executores é aproximadamente uniforme e bem abaixo do limite de segurança para todos os executores.



Monitorar o progresso de vários trabalhos

Você pode criar o perfil de vários trabalhos do AWS Glue e monitorar o fluxo de dados entre eles. Esse é um padrão de fluxo de trabalho comum e requer monitoramento de andamento de trabalho

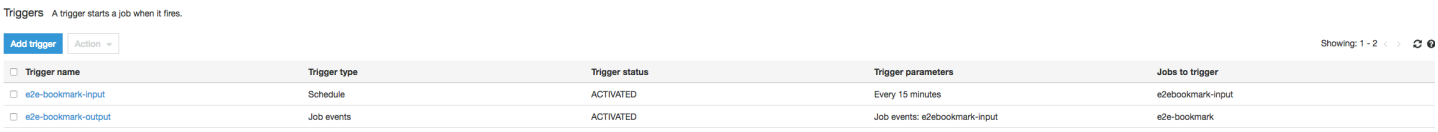
individual, backlog de processamento de dados, reprocessamento de dados e marcadores de trabalho.

Tópicos

- Código perfilado
- Visualizar as métricas perfiladas no console do AWS Glue
- Corrigir o processamento de arquivos

Código perfilado

Neste fluxo de trabalho, você tem dois trabalhos: um de entrada e um de saída. O trabalho de entrada está programado para ser executado a cada 30 minutos usando um gatilho periódico. O trabalho de saída está programado para ser executado após cada execução bem-sucedida do trabalho de entrada. Essas tarefas agendadas são controladas usando gatilhos de trabalho.



Trabalho de entrada: esse trabalho lê dados de um local do Amazon Simple Storage Service (Amazon S3), transforma-os usando `ApplyMapping` e os grava em um local de preparação do Amazon S3. O código a seguir é o código perfilado para o trabalho 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")
```

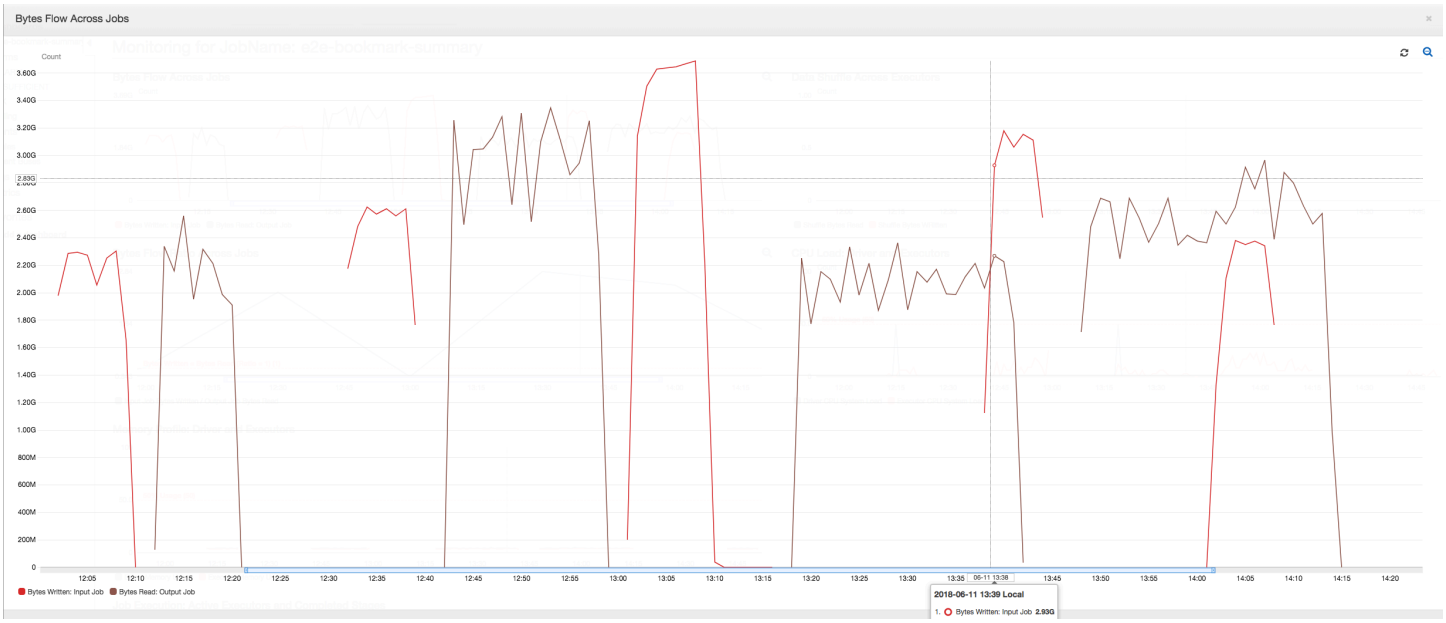
Trabalho de saída: esse trabalho lê a saída do trabalho de entrada no local de preparação no Amazon S3, transforma-a novamente e a grava em um destino:

```
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 as métricas perfiladas no console do AWS Glue

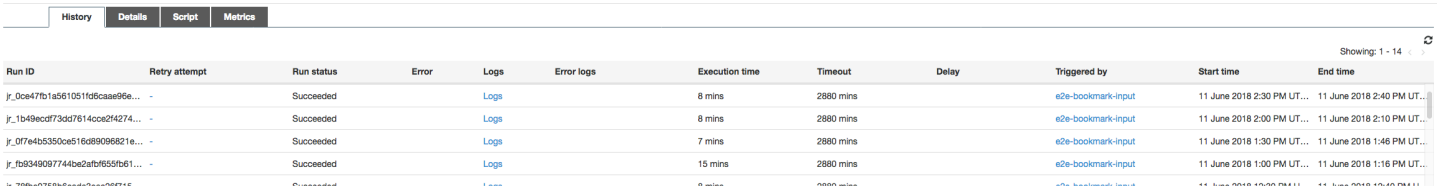
O seguinte painel sobrepõe a métrica de bytes gravados do Amazon S3 do trabalho de entrada na métrica de bytes lidos do Amazon S3 no mesmo cronograma do trabalho de saída. O cronograma mostra diferentes execuções de trabalhos de entrada e saída. O trabalho de entrada (mostrado em vermelho) começa a cada 30 minutos. O trabalho de saída (mostrado em marrom) começa na conclusão do trabalho de entrada, com uma simultaneidade máxima de 1.



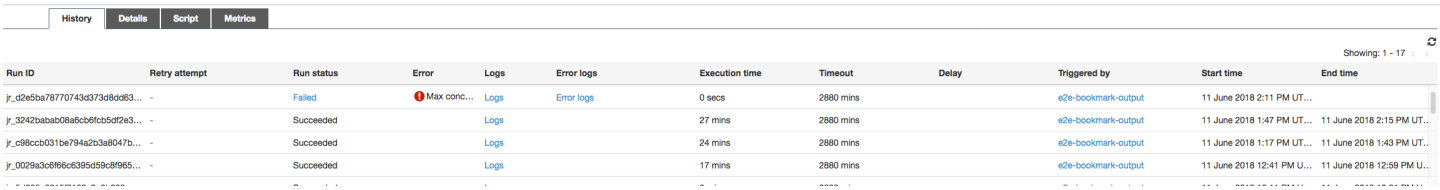
Neste exemplo, marcadores de trabalho não estão habilitados. Nenhum contexto de transformação é usado para habilitar marcadores de trabalho no código de script.

Histórico de trabalho: os trabalhos de entrada e saída têm várias execuções, conforme mostrado na guia History (Histórico), a partir do meio-dia.

O trabalho de entrada no console do AWS Glue é semelhante a este:

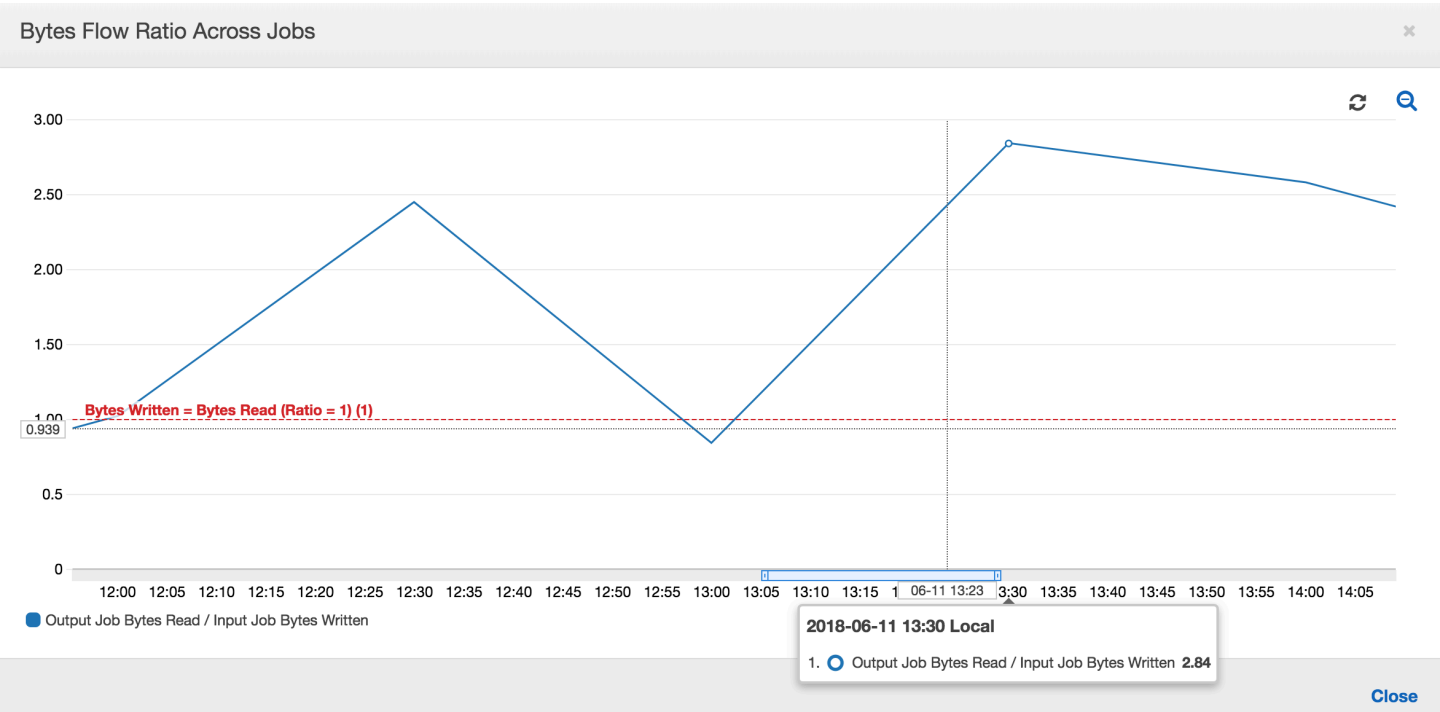


A imagem a seguir mostra o trabalho de saída:



Primeiras execuções de trabalho: como mostrado no gráfico de bytes de dados lidos e gravados abaixo, as primeiras execuções dos trabalhos de entrada e saída entre 12h e 12h30 mostram aproximadamente a mesma área sob as curvas. Essas áreas representam os bytes do Amazon S3 gravados pelo trabalho de entrada e os bytes do Amazon S3 lidos pelo trabalho de saída. Esses dados também são confirmados pela relação de bytes do Amazon S3 gravados (somados por 30 minutos: a frequência do acionador do trabalho de entrada). O ponto de dados da relação para a execução do trabalho de entrada que começou ao meio-dia também é 1.

O gráfico a seguir mostra a relação do fluxo de dados em todas as execuções de trabalho:



Segunda execução do trabalho: na segunda execução do trabalho, existe uma clara diferença no número de bytes lidos pelo trabalho de saída em comparação com o número de bytes gravados pelo trabalho de entrada. Compare a área sob a curva nas duas execuções do trabalho de saída, ou compare as áreas na segunda execução dos trabalhos de entrada e saída. A relação dos bytes lidos e gravados mostra que o trabalho de saída lê cerca de 2,5x os dados gravados pelo trabalho de entrada no segundo período de 30 minutos das 12h30 às 13h. Isso ocorre porque o trabalho de saída reprocessou a saída da primeira execução do trabalho de entrada porque os marcadores de

trabalho não foram ativados. Uma relação acima de 1 mostra que há um backlog adicional de dados que foi processado pelo trabalho de saída.

Terceira execução do trabalho: o trabalho de entrada é bastante consistente em termos do número de bytes gravados (consulte a área sob as curvas vermelhas). No entanto, a terceira execução do trabalho de entrada foi maior que o esperado (consulte a extremidade longa da curva vermelha). Como resultado, a terceira execução do trabalho de saída iniciou com atraso. A terceira execução do trabalho processou apenas uma fração dos dados acumulados no local de preparação nos 30 minutos restantes entre 13h e 13h30. A relação do fluxo de bytes mostra que ele processou apenas 0,83 de dados gravados pela terceira execução do trabalho de entrada (consulte a relação às 13h).

Sobreposição de trabalhos de entrada e saída: a quarta execução do trabalho de entrada começou às 13h30 de acordo com a programação, antes de a terceira execução do trabalho de saída ser concluída. Existe uma sobreposição parcial entre essas duas execuções de trabalho. No entanto, a terceira execução do trabalho de saída captura apenas os arquivos listados no local de preparação do Amazon S3 quando ela começou, em torno das 13h17. Isso consiste na saída de todos os dados da primeira execução do trabalho de entrada. A taxa real às 13h30 é em torno de 2,75. A terceira execução do trabalho de saída processou cerca de 2,75x de dados gravados pela quarta execução do trabalho de entrada das 13h30 às 14h.

Como essas imagens mostram, o trabalho de saída reprocessa dados do local de preparação de todas as execuções anteriores do trabalho de entrada. Como resultado, a quarta execução do trabalho de saída é a mais longa e se sobrepõe à quinta execução do trabalho de entrada.

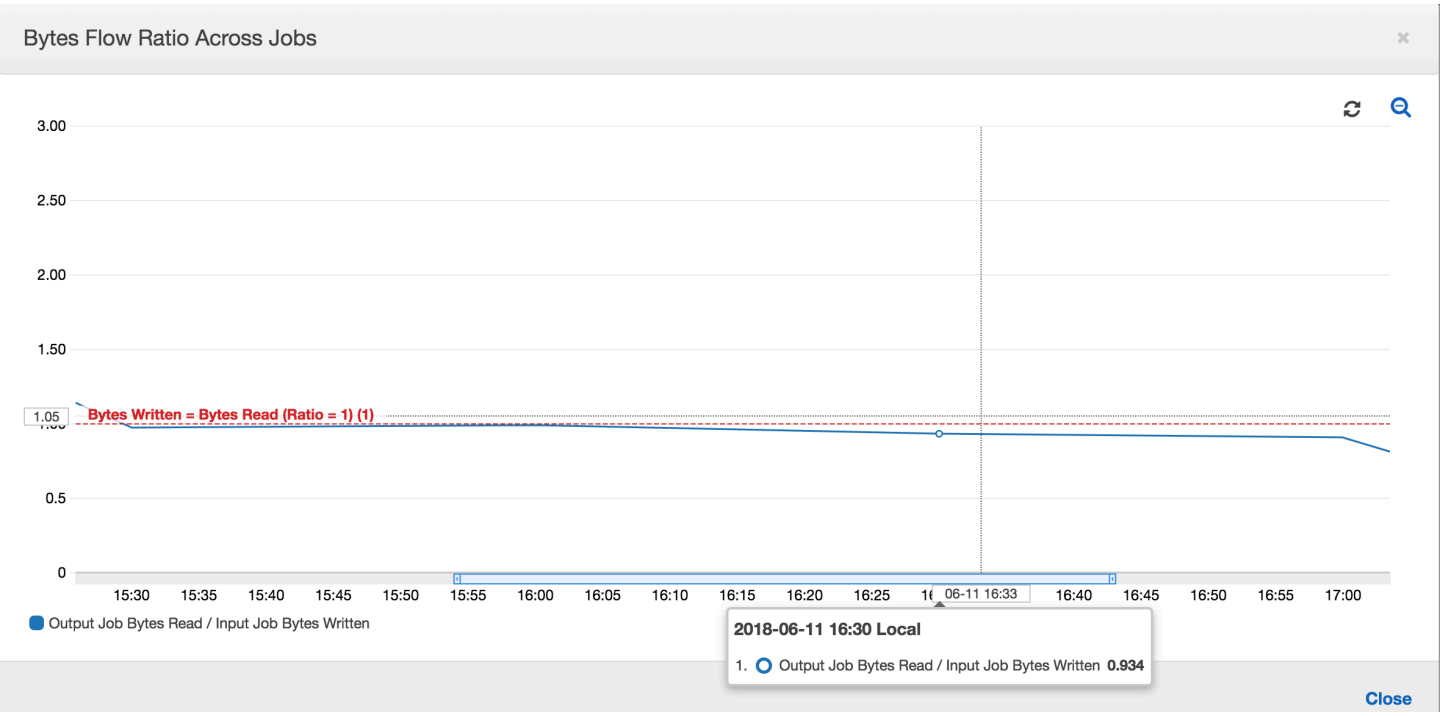
Corrigir o processamento de arquivos

Você deve garantir que o trabalho de saída processe apenas os arquivos que não foram processados por execuções anteriores do trabalho de saída. Para fazer isso, habilite marcadores de trabalho e defina o contexto de transformação no trabalho de saída, da seguinte forma:

```
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")
```

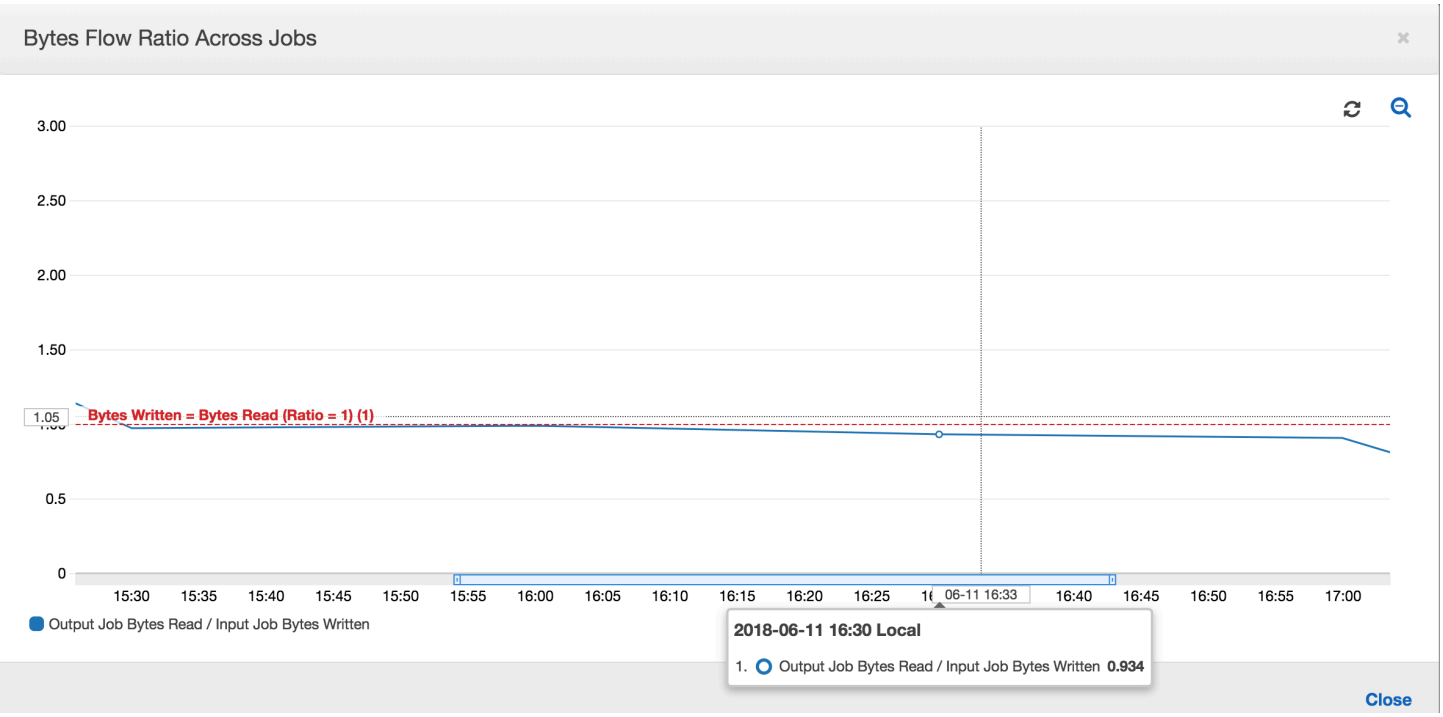
Com marcadores de trabalho ativados, o trabalho de saída não reprocessa os dados no local de preparação de todas as execuções anteriores do trabalho de entrada. Na imagem a seguir que mostra os dados lidos e gravados, a área sob a curva marrom é bastante consistente e semelhante às curvas vermelhas.

As relações de fluxo de bytes também permanecem cerca de 1 porque não há dados adicionais processados.



Uma execução do trabalho de saída é iniciada e captura os arquivos no local de preparação antes da próxima execução do trabalho de entrada começar a colocar mais dados no local de preparação.

Desde que continue a fazer isso, ela processa apenas os arquivos capturados na execução do trabalho de entrada anterior, e a relação permanece próxima a 1.



Suponha que o trabalho de entrada leve mais tempo do que o esperado e, como resultado, o trabalho de saída capture arquivos no local de preparação de duas execuções do trabalho de entrada. A relação é maior que 1 para essa execução do trabalho de saída. No entanto, as próximas execuções do trabalho de saída não processam os arquivos que já foram processados pelas execuções anteriores do trabalho de saída.

Monitorar planejamento de capacidade de DPU

Você pode usar métricas de trabalho no AWS Glue para estimar o número de unidades de processamento de dados (DPUs) que podem ser usadas para expandir um trabalho do AWS Glue.

> **Note**
>
> Esta página só é aplicável às versões 0.9 e 1.0 do AWS Glue. Versões posteriores do AWS Glue contêm recursos de economia de custos que introduzem considerações adicionais ao planejar a capacidade.

Tópicos

- Código perfilado

- Visualizar as métricas perfiladas no console do AWS Glue
- Determinar a capacidade de DPU ideal

Código perfilado

O script a seguir lê uma partição do Amazon Simple Storage Service (Amazon S3) contendo 428 arquivos JSON em gzip. O script aplica um mapeamento para alterar os nomes dos campos e os converte e grava no Amazon S3 no formato Apache Parquet. Você provisiona dez DPUs de acordo com o padrão e executa esse trabalho.

```
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")
```

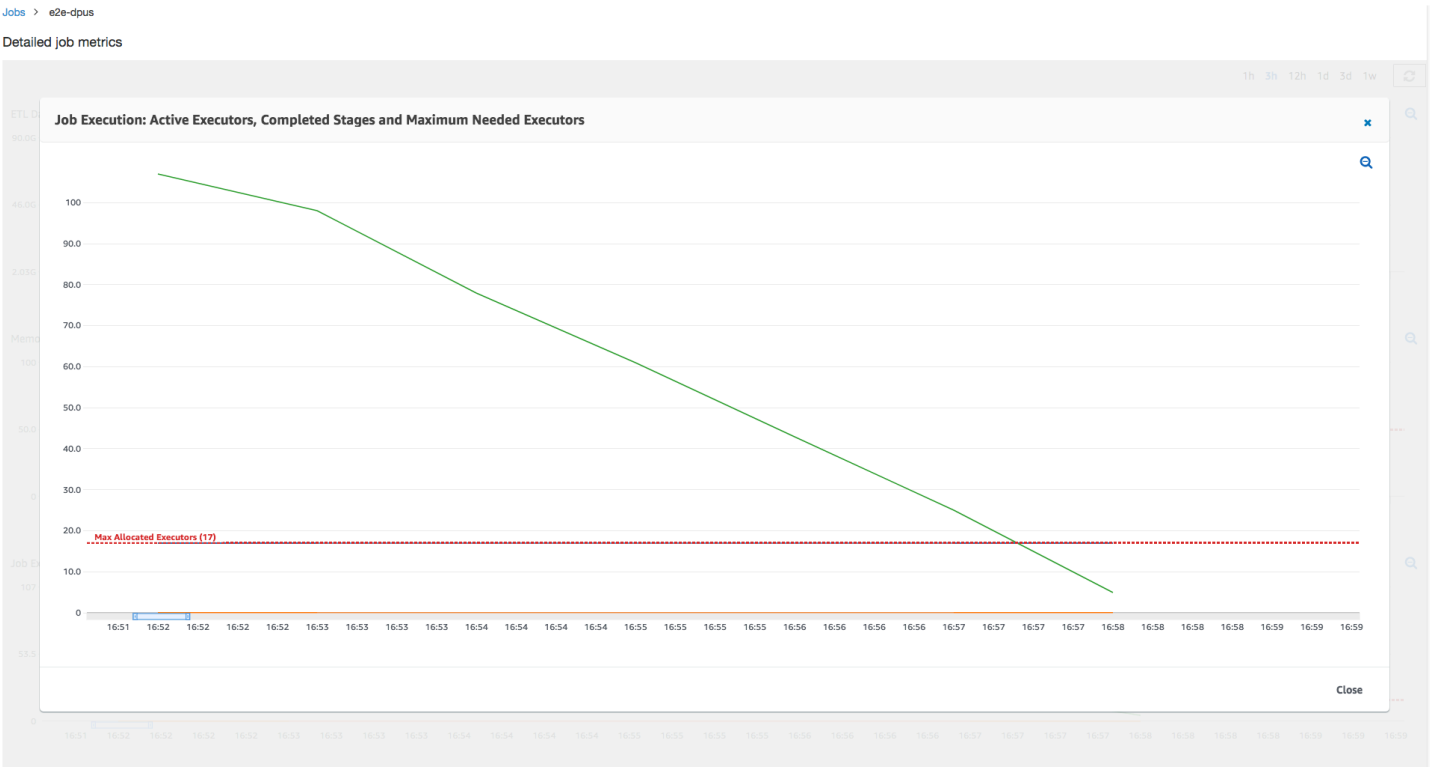
Visualizar as métricas perfiladas no console do AWS Glue

Execução de trabalho 1: nessa execução de trabalho, mostramos como descobrir se há DPUs não provisionadas no cluster. A funcionalidade da execução do trabalho no AWS Glue mostra o número total de executores ativamente em execução, o número de estágios concluídos e o número máximo de executores necessários.

O número máximo de executores necessário é calculado adicionando o número total de tarefas em execução e tarefas pendentes e dividindo pelas tarefas por executor. Esse resultado é uma medida do número total de executores necessários para satisfazer a carga atual.

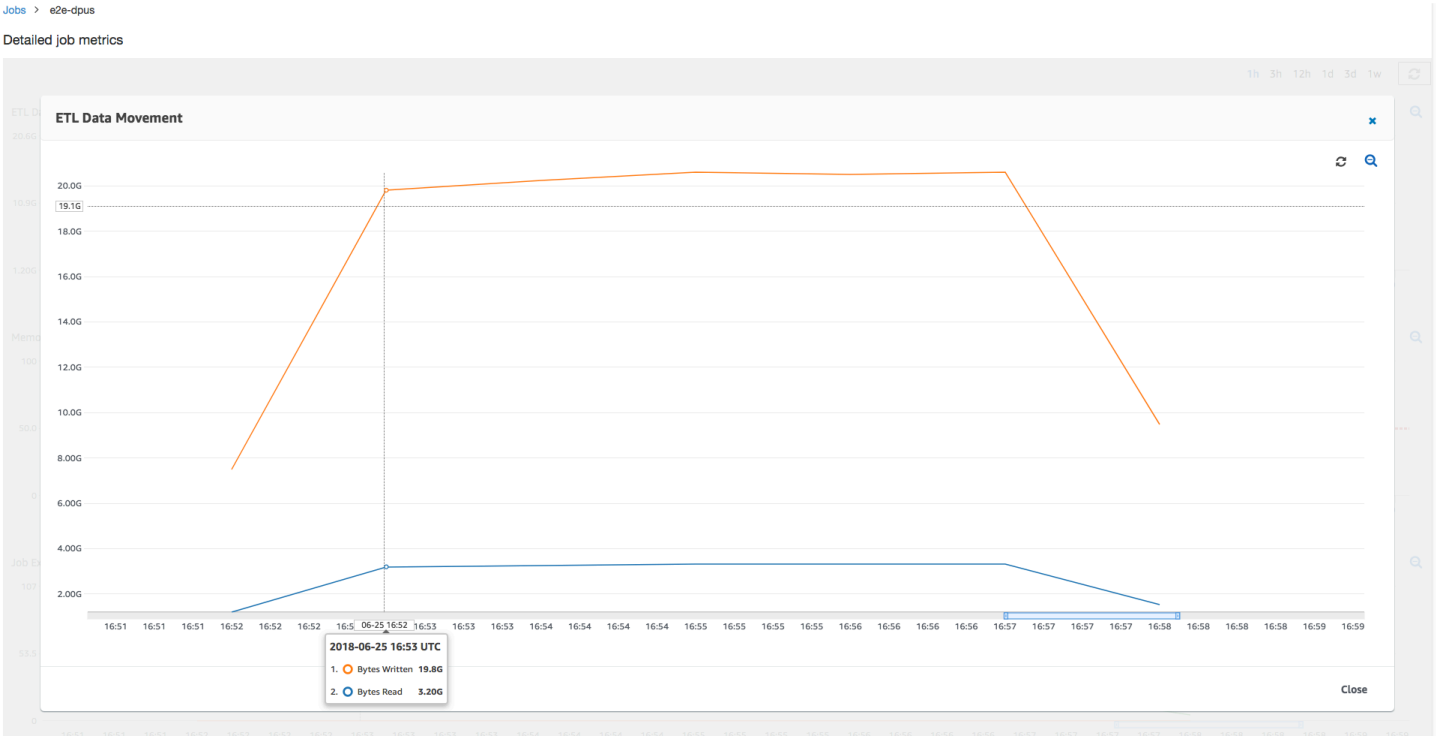
Por outro lado, o número de executores em execução ativamente mede quantos executores estão executando tarefas ativas do Apache Spark. À medida que o trabalho progride, o máximo necessário de executores pode mudar e normalmente diminui no final do trabalho conforme a fila de tarefas pendentes fica menor.

A linha vermelha horizontal no gráfico a seguir mostra o número de executores máximos alocados, que depende do número de DPUs que você aloca para o trabalho. Nesse caso, você aloca 10 DPUs para a execução do trabalho. Uma DPU é reservada para gerenciamento. Nove DPUs executam dois executores cada, e um executor é reservado para o driver do Spark. O driver do Spark é executado dentro da aplicação principal. Portanto, o número máximo de executores alocados é 2*9 - 1 = 17 executores.



Como o gráfico mostra, o número de executores máximos necessários começa em 107 no início do trabalho, enquanto o número de executores ativos permanece 17. Isso equivale ao número de executores máximos alocados com 10 DPUs. A relação entre o máximo de executores necessários e o máximo de executores alocados (adicionando 1 a ambos para o driver do Spark) fornece o fator de subprovisionamento: 108/18 = 6x. É possível provisionar 6 (de acordo com a taxa de provisionamento) * 9 (capacidade atual de DPU - 1) + 1 DPUs = 55 DPUs para expandir o trabalho a fim de executá-lo com o máximo de paralelismo e terminar mais rapidamente.

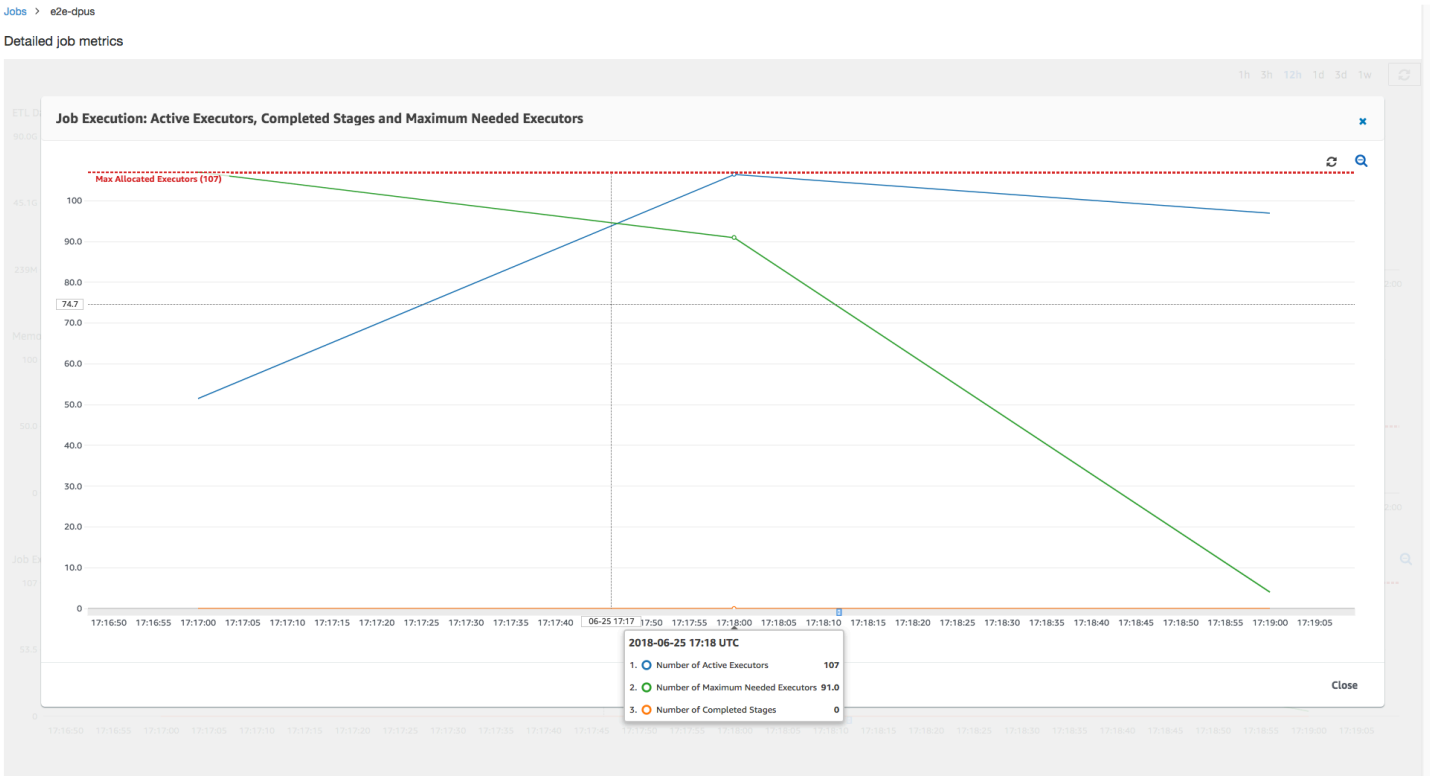
O console do AWS Glue exibe as métricas de trabalho detalhadas como uma linha estática representando o número original do máximo de executores alocados. O console computa o máximo de executores alocados da definição do trabalho para as métricas. Em contrapartida, para as métricas detalhadas de execução do trabalho, o console computa o máximo de executores alocados da configuração de execução de trabalho, especificamente as DPUs alocadas para a execução do trabalho. Para visualizar as métricas da execução de um trabalho individual, selecione a execução do trabalho e selecione View run metrics (Visualizar métricas de execução).



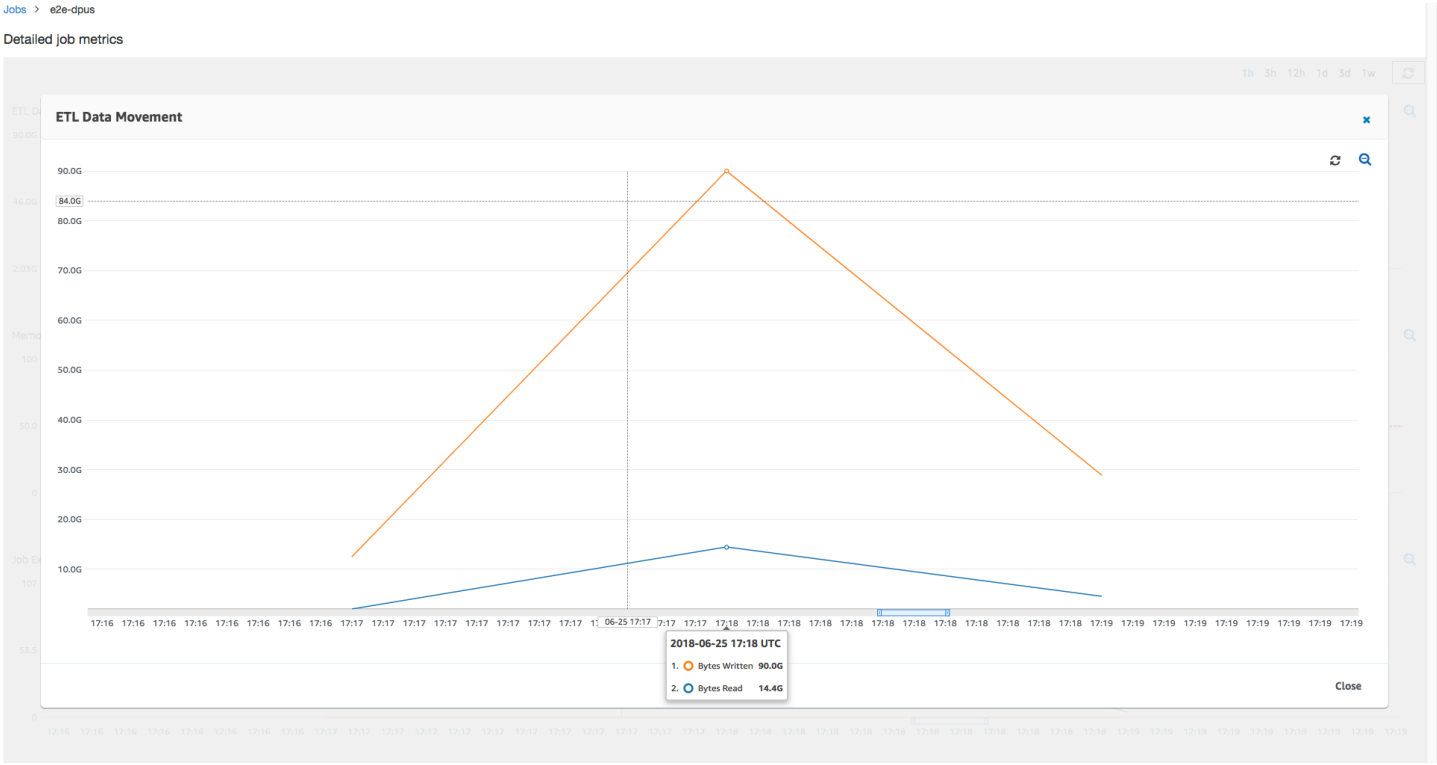
Analisando os bytes lidos e gravados do Amazon S3, observe que o trabalho gasta os seis minutos de transmissão em dados do Amazon S3 e na gravação simultaneamente. Todos os núcleos nas DPUs alocadas estão lendo e gravando no Amazon S3. O número máximo de executores necessários sendo 107 também corresponde ao número de arquivos no caminho de entrada do Amazon S3: 428. Cada executor pode iniciar quatro tarefas do Spark para processar quatro arquivos de entrada (JSON gzip).

Determinar a capacidade de DPU ideal

Com base nos resultados da execução de trabalho anterior, você pode aumentar o número total de DPUs alocadas para 55 e ver como o trabalho é realizado. O trabalho é concluído em menos de três minutos, metade do tempo necessário anteriormente. A expansão do trabalho não é linear nesse caso porque ele é um trabalho de execução curta. Os trabalhos com tarefas de longa duração ou um grande número de tarefas (um grande número de executores máximos necessários) se beneficiam de uma aceleração da performance de expansão da DPU quase linear.



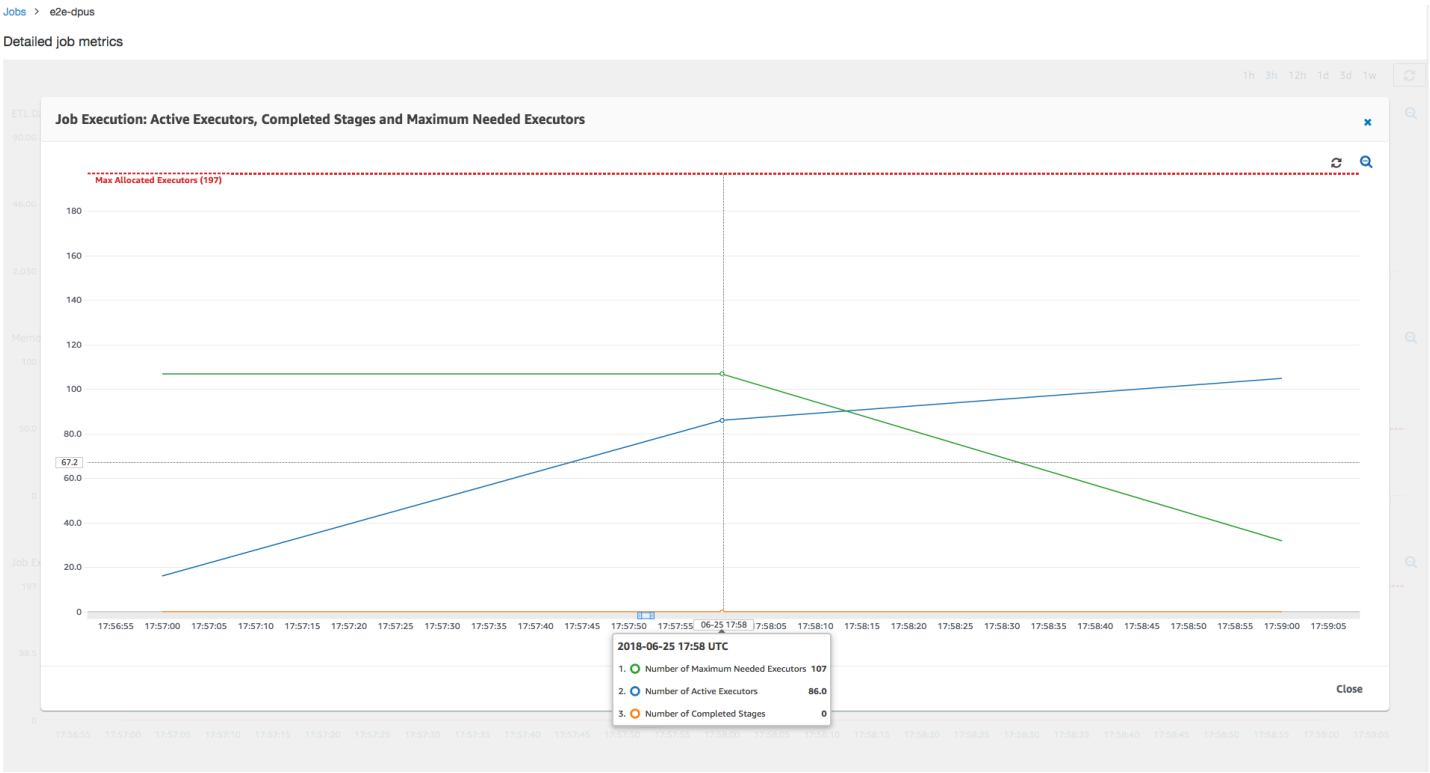
Como mostra a imagem a seguir, o número total de executores ativos atinge o máximo alocado: 107 executores. Da mesma forma, o máximo necessário de executores nunca é acima do máximo alocado de executores. O número máximo de executores necessários é calculado a partir da contagem de tarefas pendentes e ativamente em execução e, portanto, pode ser menor do que o número de executores ativos. Isso ocorre porque pode haver executores que estão parcial ou completamente ociosos por um curto período e ainda não estão desativados.



Essa execução de trabalho usa seis vezes mais executores para ler e gravar no Amazon S3 simultaneamente. Como resultado, essa execução de trabalho usa mais largura de banda do Amazon S3 para leituras e gravações e termina mais rapidamente.

Identificar DPUs provisionadas em excesso

Em seguida, você pode determinar se expandir o trabalho com 100 DPUs (99 * 2 = 198 executores) ajuda a expandir mais. Como o seguinte gráfico mostra, o trabalho ainda tem três minutos para terminar. Da mesma forma, o trabalho não ser expandido além de 107 executores (configuração de 55 DPUs) e os 91 executores restantes são fornecidos em excesso e não são usados. Isso mostra que o aumento do número de DPUs nem sempre pode melhorar a performance, como evidente no máximo necessário de executores.



Comparar diferenças de tempo

As três execuções de trabalho mostradas na tabela a seguir resumem os tempos de execução de trabalho para 10 DPUs, 55 DPUs e 100 DPUs. Você pode ver que a capacidade da DPU melhora o tempo de execução do trabalho usando as estimativas estabelecidas ao monitorar a primeira execução do trabalho.

ID do trabalho	Número de DPUs	Tempo de execução
jr_c894524c8ef5048a4d9...	10	6 min.
jr_1a466cf2575e7ffe6856...	55	3 min.
jr_34fa1ed4c6aa9ff0a814...	100	3 min.

Trabalhos de transmissão de ETL no AWS Glue

É possível criar trabalhos de extração, transformação e carregamento (ETL) de transmissão que sejam executados continuamente e que consumam dados de fontes de transmissão, como o Amazon Kinesis Data Streams, Apache Kafka e Amazon Managed Streaming for Apache Kafka

(Amazon MSK). Os trabalhos limpam e transformam os dados e, em seguida, carregam os resultados em data lakes do Amazon S3 ou armazenamentos de dados JDBC.

Além disso, você pode produzir dados para streams do Amazon Kinesis Data Streams. Esse recurso só está disponível ao escrever AWS Glue scripts. Para ter mais informações, consulte the section called “Conexões do Kinesis”.

Por padrão, o AWS Glue processa e grava dados em janelas de 100 segundos. Isso permite que os dados sejam processados de forma eficiente e que as agregações sejam realizadas em dados que chegam mais tarde do que o esperado. É possível modificar esse tamanho da janela para aumentar a pontualidade ou a precisão da agregação. Trabalhos de transmissão do AWS Glue usam pontos de verificação em vez de marcadores de trabalho para rastrear os dados que foram lidos.

> **Note**
>
> O AWS Glue cobra por hora por trabalhos de ETL de transmissão enquanto eles estão em execução.

Este vídeo discute os desafios de custo do ETL de streaming e os recursos de economia de custos em. AWS Glue

A criação de um trabalho de ETL de streaming envolve as seguintes etapas:

1. Para uma fonte de transmissão Apache Kafka, crie um conexão do AWS Glue com a origem do Kafka ou o cluster do Amazon MSK.
2. Crie manualmente uma tabela do Data Catalog para a fonte de transmissão.
3. Crie um trabalho de ETL para a fonte de dados de streaming. Defina propriedades de trabalho específicas de streaming e forneça seu próprio script ou, opcionalmente, modifique o script gerado.

Para ter mais informações, consulte ETL de streaming no AWS Glue.

Ao criar um trabalho de ETL de transmissão para o Amazon Kinesis Data Streams, você não precisa criar uma conexão do AWS Glue. No entanto, se houver uma conexão anexada ao trabalho de ETL de transmissão do AWS Glue que tenha o Kinesis Data Streams como origem, será necessário um endpoint de nuvem privada virtual (VPC) para o Kinesis. Para mais informações, consulte Criar um endpoint de interface no Guia do usuário da Amazon VPC. Ao especificar uma transmissão

do Amazon Kinesis Data Streams em outra conta, você deve configurar as funções e políticas para permitir o acesso entre contas. Para obter mais informações, consulte Exemplo: Ler de uma transmissão do Kinesis em outra conta.

Os trabalhos de ETL do AWS Glue podem detectar automaticamente dados compactados, descompactar de forma transparente os dados de fluxo, realizar as transformações usuais na fonte de entrada e carregar no armazenamento de saída.

O AWS Glue oferece suporte à descompactação automática para os seguintes tipos de compactação, considerando o formato de entrada:

Tipo de compactação	Arquivo Avro	Dado Avro	JSON	CSV	Grok
BZIP2	Sim	Sim	Sim	Sim	Sim
GZIP	Não	Sim	Sim	Sim	Sim
SNAPPY	Sim (Snappy bruto)	Sim (Snappy com enquadramento)	Sim (Snappy com enquadramento)	Sim (Snappy com enquadramento)	Sim (Snappy com enquadramento)
XZ	Sim	Sim	Sim	Sim	Sim
ZSTD	Sim	Não	Não	Não	Não
DEFLATE	Sim	Sim	Sim	Sim	Sim

Tópicos

- Criar uma conexão do AWS Glue para um fluxo de dados do Apache Kafka
- Criar uma tabela do Data Catalog para uma fonte de transmissão
- Notas e restrições para fontes de transmissão Avro
- Aplicar padrões grok a fontes de transmissão
- Definir propriedades de trabalho para um trabalho de ETL de transmissão
- Notas e restrições sobre ETL de transmissão

Criar uma conexão do AWS Glue para um fluxo de dados do Apache Kafka

Para ler a partir de um stream do Apache Kafka, é necessário criar uma conexão do AWS Glue.

Como criar uma conexão do AWS Glue para uma fonte do Kafka (console)

1. Abra o AWS Glue console em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em Data catalog (catálogo de dados), selecione Connections (Conexões).
3. Escolha Add connection (Adicionar conexão) e, na página Set up your connection's properties (Configurar propriedades da conexão) insira um nome de conexão.

   > **Note**
   >
   > Para obter mais informações sobre como especificar propriedades de conexão, consulte Propriedades de conexão do AWS Glue.

4. Em Tipo de conexão, escolha Kafka.
5. Para os URLs de servidores de bootstrap do Kafka, insira o host e o número da porta para o agente de bootstrap do cluster do Amazon MSK ou do Apache Kafka. Use somente endpoints de Transport Layer Security (TLS) para estabelecer a conexão inicial com o cluster do Kafka. Os endpoints plaintext não são compatíveis.

   Veja a seguir um exemplo de lista de pares de nome de host e de números de porta para um cluster do Amazon MSK.

   ```
   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 mais informações sobre como obter as informações do agente de bootstrap, consulte Obter os agentes de bootstrap para um cluster do Amazon MSK no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.
6. Se você quiser uma conexão segura com a origem dos dados do Kafka, selecione Require SSL connection (Exigir conexão SSL), e em Kafka private CA certificate location (Localização do certificado CA privado do Kafka), insira um caminho válido do Amazon S3 para um certificado SSL personalizado.

Para uma conexão SSL com o Kafka autogerenciado, o certificado personalizado é obrigatório. Ele é opcional para o Amazon MSK.

Para obter mais informações sobre como especificar um certificado personalizado para o Kafka, consulte the section called “Propriedades de conexão SSL”.

7. Use AWS Glue Studio ou a AWS CLI para especificar um método de autenticação do cliente Kafka. Para acessar, AWS Glue Studio AWS Glueselecione no menu ETL no painel de navegação esquerdo.

   Para obter mais informações sobre métodos de autenticação do cliente Kafka, consulte Propriedades de conexão do AWS Glue Kafka para autenticação do cliente .
8. Opcionalmente, insira uma descrição e, em seguida, escolha Next (Próximo).
9. Para um cluster do Amazon MSK, especifique sua nuvem privada virtual (VPC), sub-rede e grupo de segurança. As informações da VPC são opcionais para o Kafka autogerenciado.
10. Selecione Next (Próximo) para rever todas as propriedades de conexão e, em seguida, escolha Finish (Finalizar).

Para obter mais informações sobre conexões do AWS Glue, consulte Conectar a dados.

Propriedades de conexão do AWS Glue Kafka para autenticação do cliente

Autenticação SASL/GSSAPI (Kerberos)

Escolher esse método de autenticação permitirá que você especifique propriedades Kerberos.

Kerberos Keytab

Escolha o local do arquivo keytab. Um keytab armazena chaves de longo prazo para um ou mais principais. Para obter mais informações, consulte Documentação do MIT Kerberos: Keytab.

Arquivo Kerberos krb5.conf

Escolha o arquivo krb5.conf. Ele contém o realm padrão (uma rede lógica, semelhante a um domínio, que define um grupo de sistemas sob o mesmo KDC) e a localização do servidor KDC. Para obter mais informações, consulte Documentação do MIT Kerberos: krb5.conf.

Entidade principal do Kerberos e nome do serviço Kerberos

Insira a entidade principal do Kerberos e nome do serviço. Para obter mais informações, consulte Documentação do MIT Kerberos: entidade principal do Kerberos.

Autenticação SASL/SCRAM-SHA-512

Escolher esse método de autenticação permitirá que você especifique credenciais de autenticação.

AWS Secrets Manager

Pesquise seu token na caixa Pesquisar digitando o nome ou o ARN.

Nome de usuário e senha do provedor diretamente

Pesquise seu token na caixa Pesquisar digitando o nome ou o ARN.

Autenticação de cliente SSL

Escolher esse método de autenticação permite que você selecione o local do keystore do cliente Kafka ao navegar no Amazon S3. Opcionalmente, você pode inserir a senha do keystore do cliente Kafka e a senha da chave do cliente Kafka.

Autenticação do IAM

O método de autenticação não requer especificações adicionais e só é aplicável quando a fonte do streaming é o MSK Kafka.

Autenticação SASL/PLAIN

Escolher esse método de autenticação permite a você especificar credenciais de autenticação.

Criar uma tabela do Data Catalog para uma fonte de transmissão

É possível criar manualmente uma tabela do Data Catalog que especifique as propriedades do fluxo de dados da fonte, incluindo o esquema de dados. Essa tabela é usada como fonte de dados para o trabalho de ETL de streaming.

Se você não souber o esquema dos dados no fluxo de dados de origem, poderá criar a tabela sem um esquema. Em seguida, quando você cria o trabalho ETL de transmissão, pode ativar a função de detecção de esquema do AWS Glue. O AWS Glue determina o esquema dos dados de transmissão.

Use o AWS Glueconsole, o AWS Command Line Interface (AWS CLI) ou a AWS Glue API para criar a tabela. Para obter informações sobre como criar uma tabela manualmente com o console do AWS Glue, consulte the section called “Criar tabelas”.

> **Note**
>
> Você não pode usar o AWS Lake Formation console para criar a tabela; você deve usar o AWS Glue console.

Considere também as seguintes informações para fontes de transmissão no formato Avro ou para dados de log aos quais você pode aplicar padrões Grok.

- the section called “Notas e restrições para fontes de transmissão Avro”
- the section called “Aplicar padrões grok a fontes de transmissão”

Tópicos

- Fonte de dados do Kinesis
- Fonte de dados do Kafka
- Fonte da tabela do registro de esquemas do AWS Glue

Fonte de dados do Kinesis

Ao criar a tabela, defina as seguintes propriedades de ETL de transmissão (console).

Tipo de fonte

Kinesis

Para uma fonte do Kinesis na mesma conta:

Região

A AWS região em que o serviço Amazon Kinesis Data Streams reside. O nome da transmissão do Kinesis e da região são traduzidos juntos para um ARN de transmissão.

Exemplo: https://kinesis.us-east-1.amazonaws.com

Nome da transmissão do Kinesis

Nome do stream conforme descrito em Criar um stream no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Para obter uma fonte do Kinesis em outra conta, consulte este exemplo para configurar as funções e políticas a fim de permitir o acesso entre contas. Defina estas configurações:

ARN da transmissão

O ARN do fluxo de dados do Kinesis no qual o consumidor está registrado. Para obter mais informações, consulte Amazon Resource Names (ARNs) e AWS Service Namespaces no. Referência geral da AWS

ARN da função assumida

O nome do recurso da Amazon (ARN) da função a ser assumida.

Nome da sessão (opcional)

Um identificador para a sessão de função assumida.

Use o nome da sessão de função para identificar exclusivamente uma sessão quando a mesma função for assumida por outras entidades principais ou por motivos diferentes. Em cenários entre contas, o nome da sessão de função fica visível e pode ser acessado pela conta que possui a função. O nome da sessão de função também é usado no ARN da entidade principal da função assumida. Isso significa que as solicitações subsequentes da API entre contas que usam as credenciais de segurança temporárias exporão o nome da sessão da função à conta externa em seus registros. AWS CloudTrail

Para definir propriedades de ETL de transmissão para o Amazon Kinesis Data Streams (API do AWS Glue ou AWS CLI)

- Para configurar as propriedades de ETL de transmissão para uma fonte do Kinesis na mesma conta, especifique os parâmetros `streamName` e `endpointUrl` na estrutura `StorageDescriptor` da operação da API `CreateTable` ou do comando da CLI `create_table`.

```
"StorageDescriptor": {
 "Parameters": {
  "typeOfData": "kinesis",
  "streamName": "sample-stream",
  "endpointUrl": "https://kinesis.us-east-1.amazonaws.com"
 }
 ...
}
```

Ou especifique o `streamARN`.

Example

```
"StorageDescriptor": {
 "Parameters": {
  "typeOfData": "kinesis",
  "streamARN": "arn:aws:kinesis:us-east-1:123456789:stream/sample-stream"
 }
 ...
}
```

- Para configurar as propriedades de ETL de transmissão para uma fonte do Kinesis na mesma conta, especifique os parâmetros `streamARN`, `awsSTSRoleARN` e `awsSTSSessionName` (opcional) na estrutura `StorageDescriptor` da operação da API `CreateTable` ou do comando da 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"
 }
 ...
}
```

Fonte de dados do Kafka

Ao criar a tabela, defina as seguintes propriedades de ETL de transmissão (console).

Tipo de fonte

Kafka

Para uma fonte do Kafka:

Nome do tópico

Nome do tópico conforme especificado no Kafka.

Conexão

Uma conexão do AWS Glue que faz referência a uma fonte do Kafka, conforme descrito em the section called “Criar uma conexão para um fluxo de dados do Kafka”.

Fonte da tabela do registro de esquemas do AWS Glue

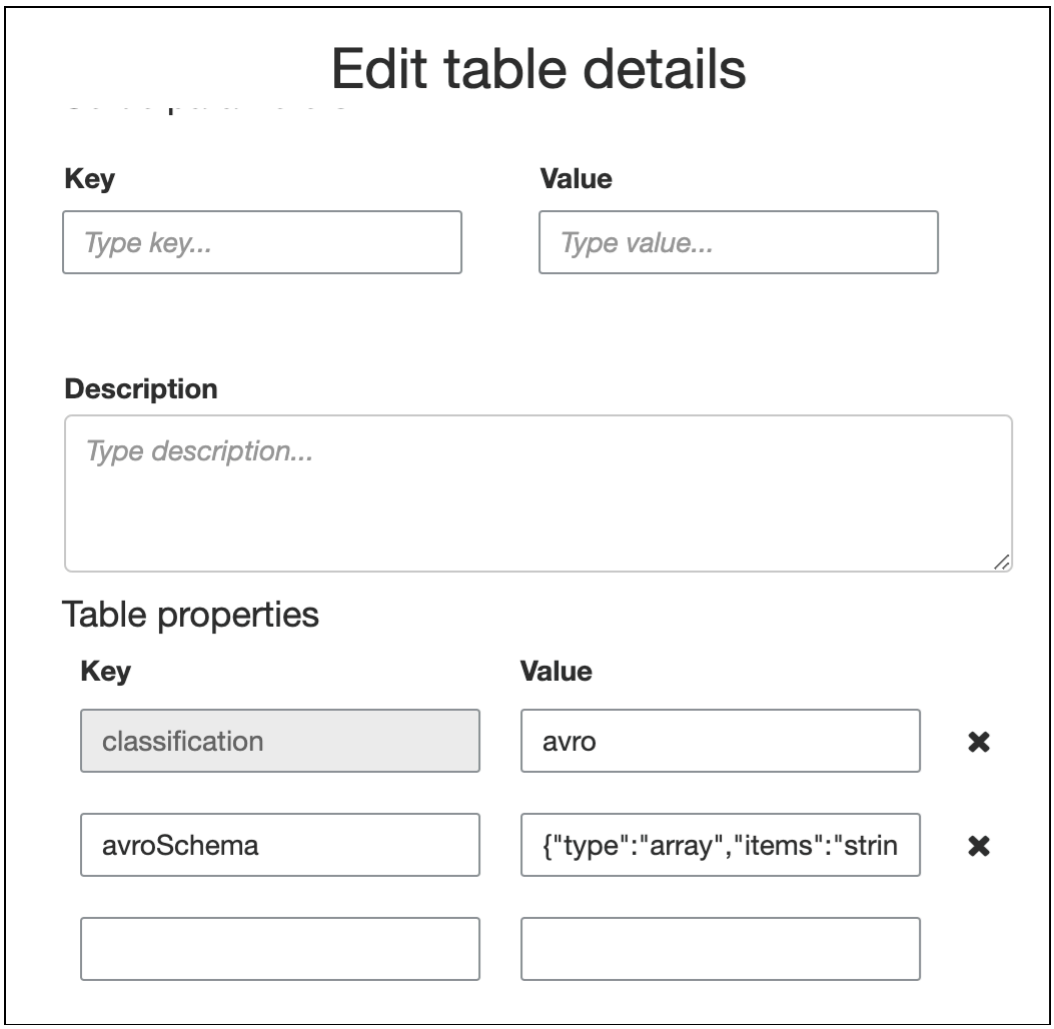
Para usar o registro de esquemas do AWS Glue para trabalhos de transmissão, siga as instruções em Caso de uso: AWS Glue Data Catalog para criar ou atualizar uma tabela de registro de esquemas.

Atualmente, a transmissão do AWS Glue oferece suporte somente ao formato Avro de registro do esquemas do Glue com inferência de esquema definida como `false`.

Notas e restrições para fontes de transmissão Avro

As seguintes notas e restrições se aplicam a fontes de transmissão no formato Avro:

- Quando a detecção de esquema está ativada, o esquema Avro deve ser incluído na carga útil. Quando desligado, a carga útil deve conter apenas dados.
- Alguns tipos de dados Avro não são suportados em quadros dinâmicos. Você não pode especificar esses tipos de dados ao definir o esquema com a página Define a schema (Definir um esquema) no assistente de criação de tabela no console do AWS Glue. Durante a detecção de esquema, os tipos não suportados no esquema Avro são convertidos para tipos suportados da seguinte forma:
  - `EnumType => StringType`
  - `FixedType => BinaryType`
  - `UnionType => StructType`
- Se você definir o esquema de tabela usando a página Define a schema (Definir um esquema) no console, o tipo de elemento raiz implícito para o esquema será `record`. Se você quiser um tipo de elemento raiz diferente de `record`, por exemplo, `array` ou `map`, não deve especificar o esquema usando a página Define a schema (Definir um esquema). Em vez disso, você deve ignorar essa página e especificar o esquema como uma propriedade de tabela ou dentro do script de ETL.
  - Para especificar o esquema nas propriedades da tabela, conclua o assistente de criação de tabela, edite os detalhes da tabela e adicione um novo par de chave-valor em Table properties (Propriedades da tabela). Use a chave `avroSchema` e insira um objeto JSON de esquema para o valor, conforme mostrado na captura de tela a seguir.



- Para especificar o esquema no script de ETL, modifique a instrução de atribuição `datasource0` e adicione a chave `avroSchema` ao argumento `additional_options`, conforme mostrado nos exemplos de Python e Scala a seguir.

  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()
```

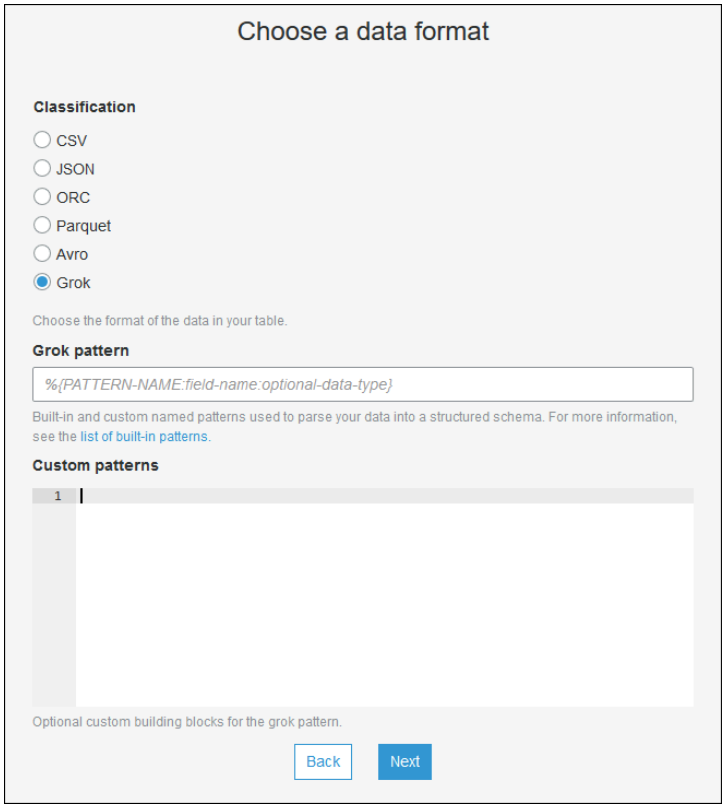
Aplicar padrões grok a fontes de transmissão

Você pode criar um trabalho de ETL de transmissão para uma fonte dos dados de log e usar padrões Grok para converter os logs em dados estruturados. Em seguida, o trabalho de ETL processa os dados como uma fonte de dados estruturada. Você especifica os padrões Grok a serem aplicados ao criar a tabela do Data Catalog para a fonte de transmissão.

Para obter informações sobre padrões Grok e valores de string padrão personalizados, consulte Gravar classificadores grok personalizados.

Para adicionar padrões grok à tabela do Data Catalog (console)

- Use o assistente de criação de tabela e a crie com os parâmetros especificados em the section called “Criar uma tabela do Data Catalog para uma fonte de transmissão”. Especifique o formato de dados como Grok, preencha o campo Grok pattern (Padrão Grok) e, opcionalmente, adicione padrões personalizados em Custom patterns (optional) (Padrões personalizados [opcional]).



  Pressione Enter após cada padrão personalizado.

Para adicionar padrões grok à tabela do Data Catalog (API do AWS Glue ou AWS CLI)

- Adicionar o parâmetro `GrokPattern` e, opcionalmente, o parâmetro `CustomPatterns` à operação da API `CreateTable` ou comando da CLI `create_table`.

```
 "Parameters": {
...
    "grokPattern": "string",
    "grokCustomPatterns": "string",
...
},
```

Expresse `grokCustomPatterns` como uma string e use “\n” como o separador entre padrões.

Veja a seguir um exemplo desses parâmetros em um arquivo.

Example

```
"parameters": {
...
    "grokPattern": "%{USERNAME:username} %{DIGIT:digit:int}",
    "grokCustomPatterns": "digit \d",
...
}
```

Definir propriedades de trabalho para um trabalho de ETL de transmissão

Ao definir um trabalho de ETL de transmissão no console do AWS Glue, forneça as seguintes propriedades específicas de transmissão. Para descrições de propriedades de trabalhos adicionais, consulte Definir propriedades de trabalho para trabalhos do Spark.

IAM role (Perfil do IAM)

Especifique a função AWS Identity and Access Management (IAM) usada para autorização de recursos usados para executar o trabalho, acessar fontes de streaming e acessar armazenamentos de dados de destino.

Para acessar o Amazon Kinesis Data Streams, `AmazonKinesisFullAccess` AWS anexe a política gerenciada à função ou anexe uma política de IAM semelhante que permita um acesso

mais refinado. Para ver políticas de exemplo, consulte Controlar o acesso aos recursos do Amazon Kinesis Data Streams usando o IAM.

Para obter mais informações sobre permissões de execução de trabalho no AWS Glue, consulte Gerenciamento de identidade e acesso para AWS Glue.

Tipo

Escolha Spark streaming (Streaming do Spark).

Versão do AWS Glue

A versão do AWS Glue determina as versões do Apache Spark, e Python ou Scala, que estão disponíveis para o trabalho. Escolha uma seleção que especifique a versão do Python ou do Scala disponível para o trabalho. O AWS Glue Versão 2.0 com suporte a Python 3 é o padrão para trabalhos de ETL de transmissão.

Janela de manutenção

Especifica uma janela na qual um trabalho de streaming pode ser reiniciado. Consulte the section called “Janelas de manutenção”.

Tempo limite de trabalho

Opcionalmente, informe uma duração em minutos. O valor padrão está em branco.

- Os trabalhos de streaming devem ter um valor de tempo limite inferior a 7 dias ou 100 a 80 minutos.
- Quando o valor for deixado em branco, o trabalho será reiniciado após 7 dias, caso você não tenha configurado uma janela de manutenção. Se você configurou uma janela de manutenção, o trabalho será reiniciado durante a janela de manutenção após 7 dias.

Fonte de dados

Remova a tabela criada em the section called “Criar uma tabela do Data Catalog para uma fonte de transmissão”.

Destino de dados

Execute um destes procedimentos:

- Escolha Create tables in your data target (Criar tabelas no destino de dados) e especifique as propriedades do destino de dados a seguir.

  Datastore

  Escolha Amazon S3 ou JDBC.

Formato

Escolha qualquer formato. Todos são compatíveis com streaming.

- Escolha Use tables in the data catalog and update your data target (Usar tabelas no catálogo de dados e atualizar seu destino de dados) e escolha uma tabela para um datastore JDBC.

Definição do esquema de saída

Execute um destes procedimentos:

- Escolha Automatically detect schema of each record (Detectar automaticamente o esquema de cada registro) para ativar a detecção de esquemas. O AWS Glue determina o esquema dos dados de transmissão.
- Escolha Specify output schema for all records (Especificar esquema de saída para todos os registros) para usar a transformação Apply Mapping (Aplicar mapeamento) e definir o esquema de saída.

Script

Opcionalmente, forneça seu próprio script ou modifique o script gerado para executar operações compatíveis com o mecanismo do Apache Spark Structured Streaming. Para obter informações sobre as operações disponíveis, consulte Operações em streaming DataFrames /conjuntos de dados.

Notas e restrições sobre ETL de transmissão

Tenha em mente as seguintes notas e restrições:

- A descompactação automática para trabalhos de ETL de fluxo do AWS Glue só está disponível para os tipos de compactação compatíveis. Observe o seguinte:
  - Snappy com enquadramento refere-se ao formato de enquadramento oficial para o Snappy.
  - Deflate é compatível com o Glue versão 3.0, mas não com o Glue versão 2.0.
- Ao usar a detecção de esquema, você não pode realizar junções de dados de transmissão.
- Os trabalhos de ETL de transmissão do AWS Glue não são compatíveis com o tipo de dados Union para registro de esquema do AWS Glue com o formato Avro.
- Seu script de ETL pode usar as transformações nativas do AWS Glue e as transformações nativas do Apache Spark Structured Streaming. Para obter mais informações, consulte Operações em streaming DataFrames /conjuntos de dados no site do Apache Spark ou. AWS Glue PySpark transforma a referência

- Os trabalhos de ETL de transmissão do AWS Glue usam pontos de verificação para acompanhar os dados que foram lidos. Portanto, um trabalho interrompido e reiniciado é retomado de onde parou no streaming. Se desejar reprocessar dados, você poderá excluir a pasta de ponto de verificação indicada no script.
- Marcadores de trabalho não são compatíveis.
- Para usar o atributo de distribuição avançada de fluxo de dados do Kinesis no trabalho, consulte the section called “Usar distribuição avançada nas tarefas de streaming do Kinesis”.
- Se você usar uma tabela de catálogo de dados criada a partir do registro de esquemas do AWS Glue, quando uma nova versão do esquema se tornar disponível, para refletir o novo esquema, você precisará fazer o seguinte:
  1. Pare os trabalhos associados à tabela.
  2. Atualize o esquema para a tabela do catálogo de dados.
  3. Reinicie os trabalhos associados à tabela.

Correspondência de registros com o FindMatches do AWS Lake Formation

> **Note**
>
> No momento, a correspondência de registros não está disponível no console do AWS Glue nas seguintes regiões: Oriente Médio (EAU), Europa (Espanha) (`eu-south-2`) e Europa (Zurique) (`eu-central-2`).

O AWS Lake Formation fornece recursos de machine learning para criar transformações personalizadas para limpar seus dados. No momento, existe uma transformação disponível chamada FindMatches. A transformação FindMatches permite identificar registros duplicados ou correspondentes no seu conjunto de dados, mesmo quando os registros não têm um identificador exclusivo comum e quando não há campos com uma correspondência exata. Isso não exigirá escrever nenhum código ou saber como o machine learning funciona. A transformação FindMatches pode ser útil em muitos problemas diferentes, como:

- Matching customers (Correspondência de clientes): vinculação de registros de clientes em diferentes bancos de dados de clientes, mesmo quando muitos campos de clientes não correspondem exatamente entre os bancos de dados (por exemplo, grafia diferente do nome, diferenças de endereço, dados ausentes ou imprecisos etc.).

- Matching products (Correspondência de produtos): correspondência de produtos em seu catálogo com relação a outras fontes de produtos, como catálogo de produtos com relação ao catálogo de um concorrente, onde as entradas são estruturadas de outra forma.
- Improving fraud detection (Melhoria da detecção de fraudes): identificação de contas de clientes duplicadas, determinação de quando uma conta recém-criada é (ou pode ser) correspondente a um usuário fraudulento conhecido anteriormente.
- Other matching problems (Outros problemas de correspondência): correspondência de endereços, filmes, listas de peças etc. Em geral, se um ser humano conseguir olhar suas linhas de banco de dados e determinar se elas são uma correspondência, haverá uma boa chance de que a transformação FindMatches possa ajudá-lo.

É possível criar essas transformações ao criar uma tarefa. A transformação que você cria tem como base um esquema de datastore de origem e dados de exemplo da fonte de dados rotulados por você (chamamos esse processo de “ensinar” uma transformação). Os registros que você rotula devem estar presentes no conjunto de dados de origem. Nesse processo, geramos um arquivo que você rotula e carrega novamente, e que ajudaria a transformação a aprender. Depois de ensinar a transformação, você poderá chamá-la no trabalho do AWS Glue baseado no Spark (PySpark ou Scala Spark) e usá-la em outros scripts com um datastore de origem compatível.

Depois que a transformação é criada, ela é armazenada no AWS Glue. No console do AWS Glue, é possível gerenciar as transformações que você cria. No painel de navegação em Integração de dados e ETL, Ferramentas de classificação de dados > Correspondência de registros, você pode editar e continuar a ensinar a sua transformação de machine learning. Para obter mais informações sobre como gerenciar transformações no console, consulte Trabalhar com transformações de machine learning no console do AWS Glue.

> **Note**
>
> AWS GlueOs trabalhos do FindMatches versão 2.0 usam o bucket `aws-glue-temp-<accountID>-<region>` do Amazon S3 para armazenar arquivos temporários enquanto a transformação está processando dados. Você pode excluir esses dados após a conclusão da execução, manualmente ou definindo uma regra de ciclo de vida do Amazon S3.

Tipos de transformações de machine learning

Você pode criar transformações de machine learning para limpar os dados. Você pode chamar essas transformações no seu script de ETL. Os seus dados passam de transformação para transformação em uma estrutura de dados chamada DynamicFrame, que é a extensão de uma `DataFrame` do Apache Spark SQL. O `DynamicFrame` contém os dados, e você referencia o esquema para processar os dados.

Os seguintes tipos de transformações de machine learning estão disponíveis:

Encontrar correspondências

Encontra registros duplicados nos dados de origem. Ensine essa transformação de machine learning rotulando exemplos de conjuntos de dados para indicar quais linhas são correspondentes. Quanto mais você ensina a transformação de machine learning com exemplos de dados rotulados, mais ela aprende quais linhas devem ser correspondências. Dependendo de como você configura a transformação, a saída será:

- Uma cópia da tabela de entrada, além de uma coluna `match_id` preenchida com valores que indicam conjuntos correspondentes de registros. A coluna `match_id` é um identificador arbitrário. Todos os registros que têm o mesmo `match_id` foram identificados como correspondentes entre si. Os registros com `match_id` diferente não são correspondentes.
- Uma cópia da tabela de entrada com as linhas duplicadas removidas. Se várias duplicatas forem encontradas, o registro com a menor chave primária será mantido.

Localização de correspondências incrementais

A transformação Find matches (Localizar correspondências) também pode ser configurada para localizar correspondências entre os quadros existentes e incrementais e retornar como saída uma coluna contendo um ID exclusivo por grupo de correspondência.

Para obter mais informações, consulte: Localizar correspondências incrementais

Usando a transformação FindMatches

É possível usar a transformação `FindMatches` para encontrar registros duplicados nos dados de origem. Um arquivo de rotulamento é gerado ou fornecido para ajudar a ensinar a transformação.

> **Note**
>
> No momento, não há suporte para transformações `FindMatches` que usem uma chave de criptografia personalizada nas seguintes regiões:
>
> - Asia Pacific (Osaka) - `ap-northeast-3`

Para começar com a transformação FindMatches, você pode seguir as etapas abaixo. Para um exemplo mais avançado e detalhado, consulte o blog de Big Data da AWS: Harmonize data using AWS Glue and AWS Lake Formation FindMatches ML to build a customer 360 view.

Conceitos básicos do uso da transformação Find Matches

Siga estas etapas para começar a usar a transformação `FindMatches`:

1. Crie uma tabela no AWS Glue Data Catalog para os dados de origem que devem ser limpos. Para obter informações sobre como criar um crawler, consulte Trabalhar com crawlers no console do AWS Glue.

   Se os seus dados de origem forem um arquivo baseado em texto, como um arquivo de valores separados por vírgulas (CSV), considere o seguinte:

   - Mantenha o arquivo CSV de registros de entrada e o arquivo de rotulamento em pastas separadas. Caso contrário, o crawler do AWS Glue poderá considerá-los como várias partes da mesma tabela e criar tabelas no Data Catalog de maneira incorreta.
   - A menos que o arquivo CSV inclua somente caracteres ASCII, garanta que a codificação UTF-8 sem BOM (marca de ordem de byte) seja usada para os arquivos CSV. Geralmente, o Microsoft Excel adiciona uma BOM no início dos arquivos CSV em UTF-8. Para removê-la, abra o arquivo CSV em um editor de texto e salve novamente o arquivo como UTF-8 without BOM (UTF-8 sem BOM).
2. No console do AWS Glue, crie uma tarefa e selecione o tipo de transformação Find matches (Encontrar correspondências).

   > **Important**
   >
   > A tabela da fonte de dados que você escolher para a tarefa não pode ter mais de 100 colunas.

3. Instrua o AWS Glue a gerar um arquivo de rotulamento selecionando Generate labeling file (Gerar arquivo de rotulamento). O AWS Glue dá o primeiro passo no agrupamento de registros semelhantes para cada `labeling_set_id`, de maneira que você possa revisar esses agrupamentos. Rotule correspondências de rótulos na coluna `label`.
   - Se você já tiver um arquivo de rotulagem, ou seja, um exemplo de registros que indicam linhas correspondentes, faça upload do arquivo no Amazon Simple Storage Service (Amazon S3). Para obter informações sobre o formato do arquivo de rotulamento, consulte Formato do arquivo de rotulagem. Prossiga para a etapa 4.
4. Faça download do arquivo de rotulamento e rotule-o conforme descrito na seção Rótulo.
5. Faça upload do arquivo de rotulamento corrigido. O AWS Glue executa tarefas para ensinar a transformação a encontrar correspondências.

   Na página da lista Machine learning transforms (Transformações de machine learning), selecione a guia History (Histórico). Essa página indica quando o AWS Glue realiza as seguintes tarefas:
   - Import labels (Importar rótulos)
   - Export labels (Exportar rótulos)
   - Generate labels (Gerar rótulos)
   - Estimate quality (Estimar qualidade)
6. Para criar uma transformação melhor, é possível fazer download, rotular e fazer upload do arquivo rotulado de maneira iterativa. Nas execuções iniciais, pode haver muito mais correspondências incorretas. No entanto, o AWS Glue aprende à medida que você continua a ensiná-lo ao verificar o arquivo de rotulamento.
7. Avalie e ajuste a transformação avaliando a performance e os resultados da localização de correspondências. Para ter mais informações, consulte Ajustar transformações de machine learning no AWS Glue.

Rótulo

Quando a `FindMatches` gera um arquivo de rotulamento, os registros são selecionados da tabela de origem. Com base no treinamento anterior, a `FindMatches` identifica os registros mais valiosos dos quais deve aprender.

A ação de rotulamento consiste na edição de um arquivo de rotulamento (sugerimos usar uma planilha como o Microsoft Excel) e na adição de identificadores, ou rótulos, na coluna `label` que identifica registros correspondentes e não correspondentes. É importante ter uma definição clara e consistente de uma correspondência nos dados de origem. A `FindMatches` aprende dos registros

que você designa como correspondências (ou não) e usa suas decisões para aprender a encontrar registros duplicados.

Quando um arquivo de rotulamento é gerado pela `FindMatches`, são gerados aproximadamente 100 registros. Geralmente, esses 100 registros são divididos em 10 conjuntos de rotulagem, nos quais cada conjunto de rótulos é identificado por um `labeling_set_id` exclusivo gerado pela `FindMatches`. Cada conjunto de rótulos deve ser visto como uma tarefa de rotulagem separada e independente dos demais conjuntos de rótulos. A sua tarefa é identificar os registros correspondentes e não correspondentes dentro de cada conjunto de rótulos.

Dicas para editar arquivos de rotulagem em uma planilha

Ao editar o arquivo de rotulamento em um aplicativo de planilha, considere o seguinte:

- O arquivo pode não abrir com os campos de coluna totalmente expandidos. Pode ser necessário expandir as colunas `labeling_set_id` e `label` para ver o conteúdo dessas células.
- Se a coluna de chave primária for um número, como um tipo de dados `long`, a planilha poderá interpretá-la como um número e alterar o valor. Esse valor de chave deve ser tratado como texto. Para corrigir esse problema, formate todas as células da coluna de chave primária como Text data (Dados de texto).

Formato do arquivo de rotulagem

O arquivo de rotulamento gerado pelo AWS Glue para ensinar a transformação `FindMatches` usa o formato a seguir. Se você gerar o seu próprio arquivo para o AWS Glue, ele também deve seguir este formato:

- É um arquivo de valores separados por vírgulas (CSV).
- Deve estar codificado em `UTF-8`. Se você editar o arquivo usando o Microsoft Windows, ele poderá ser codificado com `cp1252`.
- Ele deve estar em um local do Amazon S3 para ser transmitido ao AWS Glue.
- Use uma quantidade moderada de linhas para cada tarefa de rotulagem. Recomenda-se 10 a 20 linhas por tarefa, embora 2 a 30 linhas por tarefa seja aceitável. Tarefas com mais de 50 linhas não são recomendadas e podem gerar resultados insatisfatórios ou uma falha no sistema.
- Caso já tenha dados rotulados que consistam em pares de registros rotulados como “correspondência” ou “não correspondência”, não há problema. Esses pares rotulados podem ser representados como conjuntos de rótulos de tamanho 2. Nesse caso, rotule ambos os registros

com, por exemplo, uma letra “A” caso haja correspondência, e rotule um como “A” e outro como “B” caso não haja correspondência.

> **Note**
>
> Como ele tem colunas adicionais, o arquivo de rotulamento tem um esquema diferente de um arquivo que contém os dados de origem. Coloque o arquivo de rotulagem em uma pasta diferente de qualquer arquivo CSV de entrada da transformação, para que o crawler do AWS Glue não o considere ao criar tabelas no Data Catalog. Caso contrário, a tabela criada pelo crawler do AWS Glue poderá não representar seus dados de maneira correta.

- As duas primeiras colunas (`labeling_set_id` e `label`) são exigidas pelo AWS Glue. As colunas restantes devem corresponder ao esquema dos dados a serem processados.
- Para cada `labeling_set_id`, identifique todos os registros correspondentes usando o mesmo rótulo. Um rótulo é uma string exclusiva colocada na coluna `label`. Recomendamos o uso de rótulos que contenham caracteres simples, como A, B, C e assim por diante. Os rótulos diferenciam letras maiúsculas de minúsculas e são inseridos na coluna `label`.
- As linhas que contêm o mesmo `labeling_set_id` e o mesmo rótulo são rotuladas como uma correspondência.
- As linhas que contêm o mesmo `labeling_set_id` e um rótulo diferente são rotuladas como uma não correspondência.
- As linhas que contêm um `labeling_set_id` diferente não têm informações que confirmam ou rejeitam a correspondência.

Veja a seguir um exemplo de rotulamento dos dados:

labeling_set_id	rótulo	first_name	last_name	Aniversário
ABC123	A	John	Doe	01/04/1980
ABC123	B	Jane	Smith	03/04/1980
ABC123	A	Johnny	Doe	01/04/1980
ABC123	A	Jon	Doe	01/04/1980
DEF345	A	Richard	Jones	11/12/1992

labeling_set_id	rótulo	first_name	last_name	Aniversário
DEF345	A	Rich	Jones	12/11/1992
DEF345	B	Sara	Jones	11/12/1992
DEF345	C	Richie	Jones Jr.	06/05/2017
DEF345	B	Sara	Jones-Walker	11/12/1992
GHI678	A	Robert	Miller	03/01/1999
GHI678	A	Bob	Miller	03/01/1999
XYZABC	A	William	Robinson	05/02/2001
XYZABC	B	Andrew	Robinson	05/02/1971

- No exemplo acima, identificamos John/Johnny/Jon Doe como sendo uma correspondência e ensinamos o sistema que esses registros não correspondem a Jane Smith. Separadamente, ensinamos ao sistema que Richard e Rich Jones são a mesma pessoa, mas esses registros não correspondem a Sarah Jones/Jones-Walker e Richie Jones Jr.
- Como pode ver, o escopo dos rótulos está limitado ao `labeling_set_id`. Portanto, os rótulos não cruzam os limites de `labeling_set_id`. Por exemplo, um rótulo "A" em `labeling_set_id` 1 não tem nenhuma relação com o rótulo "A" em `labeling_set_id` 2.
- Se um registro não tiver correspondências dentro de um conjunto de rótulos, atribua um rótulo exclusivo ao registro. Por exemplo, Jane Smith não corresponde a qualquer registro no conjunto de rótulos ABC123, portanto, é o único registro nesse conjunto com o rótulo B.
- O conjunto de rótulos “GHI678” mostra que um conjunto de rótulos pode consistir em apenas dois registros, que recebem o mesmo rótulo para indicar a correspondência. Da mesma forma, “XYZABC” mostra dois registros com rótulos diferentes para indicar que não são correspondentes.
- Observe que, às vezes, um conjunto de rótulos pode não conter correspondências (ou seja, você atribui um rótulo diferente para cada registro dentro do conjunto de rótulos) ou todos os registros do conjunto de rótulos são “iguais” (todos recebem o mesmo rótulo). Isso não é um problema, contanto que os conjuntos de rótulos coletivamente contenham exemplos de registros que sejam e que não sejam “iguais” de acordo com seus critérios.

> **Important**
>
> Confirme se a função do IAM transmitida ao AWS Glue tem acesso ao bucket do Amazon S3 que contém o arquivo de rotulagem. Por convenção, as políticas do AWS Glue concedem permissão às pastas ou aos buckets do Amazon S3 cujos nomes têm como prefixo aws-glue-. Se os arquivos de rotulagem estiverem em um local diferente, adicione uma permissão a essa localização na função do IAM.

Ajustar transformações de machine learning no AWS Glue

Você pode ajustar suas transformações de machine learning no AWS Glue para aprimorar os resultados de seus trabalhos de limpeza de dados e atender aos seus objetivos. Para melhorar sua transformação, você pode ensiná-la gerando um conjunto de rotulamento, adicionando rótulos e repetindo essas etapas várias vezes até obter os resultados desejados. Você também pode ajustá-las alterando alguns parâmetros de machine learning.

Para obter mais informações sobre transformações de machine learning, consulte Correspondência de registros com o FindMatches do AWS Lake Formation.

Tópicos

Medições de machine learning

Para compreender as medições que são usadas para ajustar a transformação de machine learning, você deve estar familiarizado com a seguinte terminologia:

Verdadeiro positivo (TP)

Uma correspondência nos dados que a transformação encontrou corretamente, também denominada como um acerto.

Verdadeiro negativo (TN)

Uma falta de correspondência nos dados que a transformação rejeitou corretamente.

Falsos positivo (FP)

Uma falta de correspondência nos dados que a transformação classificou incorretamente como correspondente, também denominada como um alarme falso.

Falso negativo (FN)

Uma correspondência nos dados que a transformação não detectou, também denominada como um erro.

Para obter mais informações sobre a terminologia de machine learning, consulte Confusion matrix (em inglês) na Wikipédia.

Para ajustar as transformações de machine learning, você pode alterar o valor das seguintes medidas nas Advanced properties (Propriedades avançadas) da transformação.

- Precision (Precisão) mede o quão bem a transformação encontra verdadeiros positivos entre o número total de registros que identifica como positivos (verdadeiros positivos e falsos positivos). Para obter mais informações, consulte Precisão e revocação na Wikipédia.
- A revocação mede quão bem a transformação encontra verdadeiros positivos do total de registros nos dados de origem. Para obter mais informações, consulte Precisão e revocação na Wikipédia.
- A acurácia mede quão bem a transformação encontra verdadeiros positivos e verdadeiros negativos. Aumentar a acurácia exige mais recursos de máquina e eleva os custos. Mas isso também resulta em um aumento da revocação. Para obter mais informações, consulte Accuracy and precision (em inglês) na Wikipédia.
- O custo mede quantos recursos de computação (logo, dinheiro) são consumidos para executar a transformação.

Escolher entre precisão e recuperação

Toda transformação `FindMatches` tem um parâmetro `precision-recall`. Você usa esse parâmetro para especificar o seguinte:

- Caso esteja mais preocupado com a possibilidade da transformação detectar a correspondência de dois registros que, na verdade, não correspondem, favoreça a precisão.

- Caso esteja mais preocupado com a transformação falhar na detecção de registros que correspondem, favoreça a revocação.

Você pode ajustar esse equilíbrio no console do AWS Glue ou usando as operações da API de machine learning do AWS Glue.

Quando favorecer a precisão

Dê preferência para a precisão se você estiver mais preocupado com o risco de `FindMatches` corresponder registros que, na verdade, não correspondem. Para favorecer a precisão, escolha um valor mais alto de equilíbrio entre precisão e recall. Com um valor mais alto, a transformação `FindMatches` precisa de mais evidências para decidir se dois registros devem corresponder. A transformação será ajustada para corresponder menos registros.

Por exemplo, suponha que você esteja usando `FindMatches` para detectar itens duplicados em um catálogo de filmes e fornece um valor de precisão-revocação mais alto para a transformação. Se a sua transformação detecta incorretamente que Star Wars: Uma Nova Esperança é o mesmo que Star Wars: O Império Contra-Ataca, um cliente que procurando Uma Nova Esperança poderá ver O Império Contra-Ataca. Isso resultaria em uma experiência do cliente insatisfatória.

No entanto, se a transformação não detectar que Star Wars: Uma Nova Esperança e Star Wars: Episódio IV - Uma Nova Esperança são o mesmo item, o cliente pode ficar confuso, mas reconhecerá os dois como iguais. Isso seria um erro, mas menos prejudicial que o primeiro caso.

Quando favorecer a recuperação

Dê preferência para a revocação se você estiver mais preocupado com o risco da transformação `FindMatches` não detectar pares de registros que, na verdade, são correspondentes. Para favorecer o recall, escolha um valor mais baixo de equilíbrio entre precisão e recall. Com um valor mais baixo, a transformação `FindMatches` precisa de menos evidências para decidir se dois registros devem corresponder. A transformação será ajustada para tender a corresponder mais registros.

Por exemplo, isso pode ser uma prioridade em uma organização de segurança. Suponha que você está comparando a lista de clientes com uma lista de falsificadores, e deve determinar se cada cliente é um falsificador ou não. Você vai usar `FindMatches` para corresponder a lista de falsificadores com a lista de clientes. Sempre que `FindMatches` detectar uma correspondência entre as duas listas, um auditor humano é atribuído para verificar se aquela pessoa é realmente um falsificador. Sua organização pode preferir revocação ao invés de precisão. Ou seja, você prefere

que os auditores revisem e rejeitem manualmente os casos nos quais o cliente não é um falsificador à transformação falhar ao identificar se um cliente está na lista de falsificadores.

Como favorecer tanto a precisão como a recuperação

A melhor maneira de aprimorar a precisão e a revocação é rotulando mais dados. Conforme você rotula mais dados, a acurácia geral da transformação `FindMatches` melhora, aprimorando, por sua vez, a precisão e a revocação. No entanto, até com a maior acurácia possível em uma transformação, você sempre precisará experimentar entre favorecer precisão ou revocação, ou manter um valor equilibrado.

Decidir entre acurácia e custo

Cada transformação `FindMatches` contém um parâmetro `accuracy-cost`. Você pode usar esse parâmetro para especificar o seguinte:

- Caso esteja mais preocupado com a transformação corresponder dois registros corretamente, favoreça a acurácia.
- Caso esteja mais preocupado com o custo ou a velocidade de executar a transformação, favoreça baixo custo.

Você pode ajustar esse equilíbrio no console do AWS Glue ou usando as operações da API de machine learning do AWS Glue.

Quando favorecer a acurácia

Dê preferência para a acurácia se você estiver mais preocupado com o risco dos resultados de `find matches` não terem correspondências. Para favorecer a acurácia, escolha um valor de equilíbrio entre acurácia e custo mais alto. Com um valor mais alto, a transformação `FindMatches` precisa de mais tempo para fazer uma pesquisa mais aprofundada e corresponder registros corretamente. Observe que esse parâmetro não diminui a probabilidade de erro na correspondência de dois registros que, na verdade, não correspondem. A transformação será ajustada para gastar mais tempo detectando correspondências.

Quando favorecer o custo

Dê preferência para o custo se estiver mais preocupado com os gastos de execução da transformação `find matches` e menos com a quantidade de correspondências encontradas. Para favorecer o custo, escolha um valor de equilíbrio entre acurácia e custo mais baixo. Com um

valor mais baixo, a transformação `FindMatches` exige menos recursos para ser executada. A transformação será ajustada para tender a detectar menos correspondências. Se os resultados forem aceitáveis ao favorecer baixo custo, use essa configuração.

Como favorecer tanto a acurácia como o baixo custo

A máquina leva mais tempo para determinar se mais pares de registros correspondem. Você pode executar as seguintes ações para reduzir os gastos, mas manter a qualidade:

- Elimine registros da fonte de dados que não são relevantes para as correspondências.
- Elimine colunas da fonte de dados as quais você não tem certeza se são úteis para a identificação de correspondências. Uma boa maneira de decidir isso é eliminando as colunas que você acredita não afetarem sua própria decisão sobre um conjunto de registros ser "o mesmo".

Estimação da qualidade das correspondências usando pontuações de confiança de correspondência

As pontuações de confiança de correspondência fornecem uma estimativa da qualidade das correspondências encontradas pelo FindMatches para distinguir entre registros correspondentes nos quais o modelo de machine learning é altamente confiante, incerto ou improvável. Uma pontuação de confiança na partida ficará entre 0 e 1, onde uma pontuação mais alta significa maior semelhança. O exame das pontuações de confiança da correspondência permite distinguir entre clusters de correspondências em que o sistema é altamente confiante (que você pode decidir mesclar), clusters sobre os quais o sistema é incerto (que você pode decidir que seja revisado por um humano) e clusters que o sistema considera improváveis (que você pode decidir rejeitar).

Você pode querer ajustar seus dados de treinamento em situações em que veja uma pontuação de confiança de correspondência alta, mas determine que não há correspondências ou onde você vê uma pontuação baixa, mas determina que existem, de fato, correspondências.

As pontuações de confiança são particularmente úteis quando há conjuntos de dados industriais de grande porte, nos quais é inviável revisar todas as decisões do FindMatches.

As pontuações de confiança de correspondência estão disponíveis no AWS Glue versão 2.0 ou posterior.

Geração de pontuações de confiança de correspondência

Você pode gerar pontuações de confiança de correspondência definindo o valor booleano de `computeMatchConfidenceScores` para True (Verdadeiro) ao chamar `FindMatches` ou a API `FindIncrementalMatches`.

O AWS Glue adiciona um novo `column match_confidence_score` para a saída.

Exemplos de pontuação de correspondência

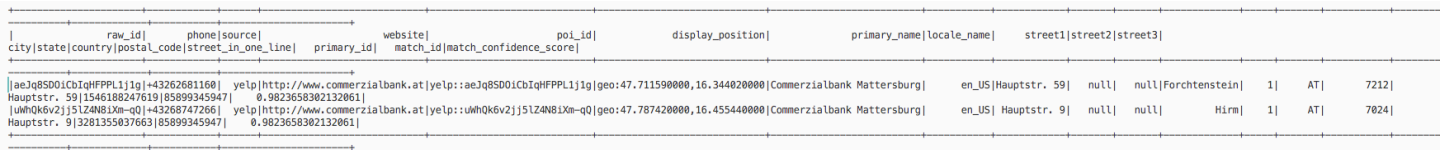
Por exemplo, considere os registros correspondentes a seguir:

Pontuação >= 0,9

Resumo dos registros correspondentes:

```
  primary_id  |   match_id  | match_confidence_score

3281355037663    85899345947   0.9823658302132061
1546188247619    85899345947   0.9823658302132061
```

Detalhes:



```
|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|
```

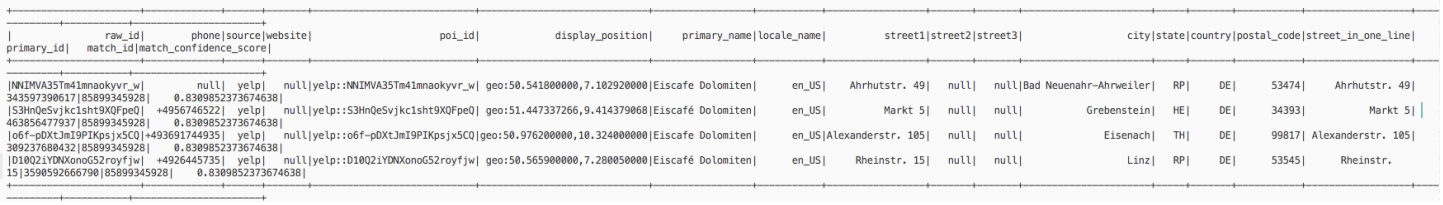
A partir deste exemplo, podemos ver que dois registros são muito semelhantes e compartilham `display_position`, `primary_name`, e `street name`.

Pontuação >= 0,8 e pontuação < 0,9

Resumo dos registros correspondentes:

```
  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
```

Detalhes:



```
|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|
```

A partir deste exemplo, podemos ver que esses registros compartilham o mesmo `primary_name`, e `country`.

Pontuação >= 0,6 e pontuação < 0,7

Resumo dos registros correspondentes:

```
  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
```

Detalhes:

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

A partir deste exemplo, podemos ver que esses registros compartilham somente o mesmo `primary_name`.

Para obter mais informações, consulte:

- Etapa 5: adicionar e executar um trabalho com sua transformação de machine learning
- PySpark: Classe FindMatches
- PySpark: Classe FindIncrementalMatches
- Scala: Classe FindMatches
- Scala: Classe FindIncrementalMatches

Ensinar a transformação Find Matches

Cada transformação `FindMatches` precisa ser ensinada para saber o que deve ou não ser considerado como correspondência. Você ensina sua transformação adicionando rótulos a um arquivo e fazendo upload de suas escolhas para o AWS Glue.

Você pode orquestrar esse rotulamento no console do AWS Glue ou usando as operações da API de machine learning do AWS Glue.

Quantas vezes devo adicionar rótulos? Quantos rótulos são necessários?

As respostas para essas perguntas dependem de você. Você deve avaliar se `FindMatches` está fornecendo o nível de acurácia de que você precisa e se o esforço adicional de rotulamento vale a pena. A melhor maneira de decidir isso é analisando as métricas “Precision (Precisão)”, “Recall (Revocação)” e “Area under the precision recall curve (Área sob a curva de precisão e revocação)” que você pode gerar selecionando Estimate quality (Estimativa de qualidade) no console do AWS Glue. Depois de rotular mais conjuntos de tarefas, execute essas métricas novamente e verifique se foram aprimoradas. Caso, depois de rotular alguns conjuntos de tarefas, você não veja uma melhoria na métrica em que está focando, a qualidade da transformação pode ter atingido seu auge.

Qual é a necessidade de ter ambos os rótulos verdadeiro positivo e verdadeiro negativo?

A transformação `FindMatches` precisa de exemplos positivos e negativos para aprender o que você considera como correspondência. Se você estiver rotulando dados de treinamento gerados por `FindMatches` (por exemplo, usando a opção I do not have labels (Eu não tenho rótulos)), a transformação `FindMatches` tentará gerar um conjunto de "label set ids" (IDs de conjunto de rótulo) para você. Em cada tarefa, você atribui o mesmo “rótulo” a alguns registros e “rótulos” diferentes a outros registros. Em outras palavras, as tarefas não costumam ser completamente iguais ou completamente diferentes (porém, não há problema se uma tarefa específica for tudo “igual” ou tudo “diferente”).

Se você estiver ensinando a sua transformação `FindMatches` usando a opção Upload labels from S3 (Fazer upload de rótulos do S3), tente incluir os exemplos de registros correspondentes e não correspondentes. Ter apenas um tipo é aceitável. Esses rótulos ajudam a aumentar a acurácia da transformação `FindMatches`, mas você ainda precisa rotular alguns registros que gerou com a opção Generate labeling file (Gerar arquivo de rotulamento).

Como posso garantir que a transformação faça correspondências conforme ensinei?

A transformação `FindMatches` aprende com os rótulos que você fornece, portanto, pode gerar pares de registros que não seguem os rótulos fornecidos. Para garantir que a transformação `FindMatches` siga seus rótulos, selecione EnforceProvidedLabels em FindMatchesParameter.

Quais técnicas podem ser usadas quando uma transformação de ML identifica como correspondentes itens que não são uma correspondência verdadeira?

Você pode usar as seguintes técnicas:

- Aumente o valor de `precisionRecallTradeoff`. Isso resultará eventualmente em menos correspondências encontradas, mas também deve dividir seu grande cluster quando atingir um valor alto o suficiente.
- Execute as linhas de saída correspondentes aos resultados incorretos e reformate-as como um conjunto de rotulamento (removendo a coluna `match_id` e adicionando uma coluna `labeling_set_id` e `label`). Se necessário, quebre-as (subdivida-as) em vários conjuntos de rotulamento para garantir que o rotulador possa ter todos os conjuntos em mente ao atribuir rótulos. Depois, rotule corretamente os conjuntos correspondentes, faça upload do arquivo de rótulo e anexe-o aos rótulos existentes. Isso pode ajudar a ensinar o transformador sobre o que deve procurar para entender o padrão.
- (Avançado) Por fim, observe os dados para ver se há um padrão que você pode detectar, mas o sistema não está observando. Pré-processe esses dados usando funções padrão do AWS Glue para normalizá-los. Destaque o que deseja que o algoritmo aprenda separando os dados que você considera importante em suas próprias colunas. Ou crie colunas combinadas de colunas cujos dados são relacionados.

Trabalhar com transformações de machine learning no console do AWS Glue

Você pode usar AWS Glue para criar transformações personalizadas de aprendizado de máquina que podem ser usadas para limpar seus dados. É possível criar essas transformações ao criar um trabalho no console do AWS Glue .

Para obter informações sobre como criar uma transformação de machine learning, consulte Correspondência de registros com o FindMatches do AWS Lake Formation.

Tópicos

- Propriedades da transformação
- Adicionar e editar transformações de machine learning
- Visualizar detalhes da transformação
- Ensine transformações usando rótulos

Propriedades da transformação

Para ver uma transformação de aprendizado de máquina existente, faça login no AWS Management Console e abra o AWS Glue console em https://console.aws.amazon.com/glue/. No painel de

navegação em Integração de dados e ETL, escolha Ferramentas de classificação de dados > Correspondência de registros.

As propriedades de cada transformação:

Nome da transformação

O nome exclusivo que você atribuiu à transformação ao criá-la.

ID

Um identificador exclusivo da transformação.

Contagem de rótulos

O número de rótulos no arquivo de rotulamento fornecido para ajudar a ensinar a transformação.

Status

Indica se a transformação está Ready (Pronta) ou Needs training (Precisa de treinamento). Para executar uma transformação de machine learning com êxito em um trabalho, ela deve estar Ready (Pronta).

Criado

A data em que a transformação foi criada.

Modificado

A data em que a transformação foi atualizada pela última vez.

Descrição

A descrição fornecida para a transformação, se uma foi fornecida.

Versão do AWS Glue

A versão do AWS Glue usado.

ID da execução

O nome exclusivo que você atribuiu à transformação ao criá-la.

Tipo de tarefa

O tipo de transformação de machine learning. Por exemplo, Find matching records (Encontrar registros correspondentes).

Status

Indica o status da execução da tarefa. Os status possíveis causas incluem:

- Starting
- Executando
- Parando
- Interrompida
- Bem-sucedida
- Com falha
- Timeout (Tempo limite)

Erro

Se o status for Falha, uma mensagem descrevendo o motivo da falha será exibida.

Adicionar e editar transformações de machine learning

É possível visualizar, excluir, configurar e ensinar ou ajustar uma transformação no console do AWS Glue. Marque a caixa de seleção ao lado da transformação na lista, selecione Action (Ação) e escolha a ação que deseja realizar.

Criar uma nova transformação de ML

Para adicionar uma nova transformação de machine learning, escolha Criar transformação. Siga as instruções no assistente Adicionar trabalho. Para ter mais informações, consulte Correspondência de registros com o FindMatches do AWS Lake Formation.

Etapa 1. Defina as propriedades da transformação.

1. Insira um nome e uma descrição (opcional).
2. Opcionalmente, defina a configuração de segurança. Consulte Usar criptografia de dados com transformações de machine learning.
3. Opcionalmente, defina as configurações de execução da tarefa. As configurações de execução de tarefas permitem que você personalize a forma como a tarefa é executada. Selecione o tipo de operador, o número de operadores, o tempo limite da tarefa (em minutos), o número de tentativas e a versão do AWS Glue.

4. Opcionalmente, defina as tags. As tags são rótulos que você pode atribuir a um AWS recurso. Cada tag consiste em uma chave e um valor opcional. As tags podem ser usadas para pesquisar e filtrar seu recurso ou rastrear seus AWS custos.

Etapa 2. Escolha a tabela e a chave primária.

1. Escolha o banco de dados e a tabela do catálogo do AWS Glue.
2. Escolha uma chave primária na tabela selecionada. A coluna da chave primária normalmente contém um identificador exclusivo para cada registro da fonte de dados.

Etapa 3. Selecione as opções de ajuste.

1. Em Recuperação versus precisão, escolha o valor de ajuste para ajustar a transformação de modo a favorecer a recuperação ou a precisão. Por padrão, Balanceado está selecionado, mas você pode optar por favorecer a recuperação ou a precisão, ou escolher Personalizado e inserir um valor entre 0,0 e 1,0 (inclusive).
2. Para Menor custo versus precisão, escolha o valor de ajuste para favorecer o menor custo ou a precisão, ou escolha Personalizado e insira um valor entre 0,0 e 1,0 (inclusive).
3. Em Aplicação de correspondência, escolha Forçar a saída a corresponder aos rótulos se quiser ensinar a transformação de ML forçando a saída a corresponder aos rótulos usados.

Etapa 4. Revisar e criar.

1. Revise as opções das etapas de 1 a 3.
2. Escolha Editar para qualquer etapa que precise ser modificada. Escolha Criar transformação para concluir o assistente de criação de transformação.

Usar criptografia de dados com transformações de machine learning

Ao adicionar uma transformação de machine learning ao AWS Glue, você pode especificar opcionalmente uma configuração de segurança associada à origem dos dados ou ao destino dos dados. Se o bucket do Amazon S3 usado para armazenar os dados for criptografado com uma configuração de segurança, especifique a mesma configuração de segurança ao criar a transformação.

Você também pode optar por usar a criptografia do lado do servidor com AWS KMS (SSE-KMS) para criptografar o modelo e os rótulos e impedir que pessoas não autorizadas os inspecionem. Se você escolher essa opção, será solicitado que escolha o nome AWS KMS key por nome ou poderá escolher Inserir um ARN de chave. Se você optar por inserir o ARN para a chave do KMS, um segundo campo será exibido onde você poderá inserir o ARN da chave do KMS.

> **Note**
>
> No momento, transformações de ML que usem uma chave de criptografia personalizada não são compatíveis com as seguintes regiões:
>
> - Asia Pacific (Osaka) - `ap-northeast-3`

Visualizar detalhes da transformação

Visualizar as propriedades da transformação

A página de propriedades da transformação inclui os atributos da sua transformação. Ela mostra os detalhes sobre a definição da transformação, incluindo o seguinte:

- Transform name (Nome da transformação) mostra o nome da transformação.
- Type (Tipo) lista o tipo da transformação.
- Status exibe se a transformação está pronta para ser usada em um script ou em uma tarefa.
- Force output to match labels (Forçar a saída para corresponder rótulos) exibe se a transformação força a saída a corresponder os rótulos fornecidos pelo usuário.
- Spark version (Versão do Spark) está relacionada à versão do AWS Glue que você escolheu em Task run properties (Propriedades de execução de tarefa) ao adicionar a transformação. O AWS Glue 1.0 e o Spark 2.4 são recomendados para a maioria dos clientes. Para obter mais informações, consulte versões do AWS Glue.

Abas Histórico, Estimativa de qualidade e Tags

Os detalhes da transformação incluem as informações definidas quando ela foi criada. Para visualizar os detalhes de uma transformação, selecione a transformação na lista Machine learning transforms (Transformações de machine learning) e revise as informações das seguintes guias:

- Histórico

- Estimar qualidade
- Tags

Histórico

A guia History (Histórico) mostra o histórico de execução das tarefas da transformação. Vários tipos de tarefas são executadas para ensinar uma transformação. Para cada tarefa, as métricas de execução incluem o seguinte:

- Run ID (ID da execução) é um identificador criado pelo AWS Glue para cada execução dessa tarefa.
- Task type (Tipo de tarefa) mostra o tipo da tarefa executada.
- Status mostra o êxito de cada tarefa listada com a execução mais recente na parte superior.
- Error (Erro) mostra os detalhes de uma mensagem de erro se a execução apresentar falha.
- Start time (Hora de início) mostra a data e a hora (hora local) em que a tarefa foi iniciada.
- Hora de término mostra a data e a hora (hora local) em que a tarefa foi concluída.
- Logs links para os logs gravados em `stdout` para esta execução de trabalho.

  O link Logs leva você ao Amazon CloudWatch Logs. Lá você pode ver os detalhes sobre as tabelas que foram criadas no AWS Glue Data Catalog e quaisquer erros encontrados. Você pode gerenciar seu período de retenção de registros no CloudWatch console. A retenção de log padrão é `Never Expire`. Para obter mais informações sobre como alterar o período de retenção, consulte Alterar a retenção de dados do registro em CloudWatch registros no Guia do usuário do Amazon CloudWatch Logs.
- Arquivo de rótulo mostra um link para o Amazon S3 de um arquivo de rótulo gerado.

Estimar qualidade

A guia Estimate quality (Estimar qualidade) mostra as métricas usadas para medir a qualidade da transformação. As estimativas são calculadas comparando as previsões de correspondência de transformação que usam um subconjunto dos dados rotulados com relação aos rótulos fornecidos. Essas estimativas são aproximadas. É possível invocar a execução de uma tarefa Estimate quality (Estimar qualidade) nessa guia.

A guia Estimate quality (Estimar qualidade) mostra as métricas da última execução de Estimate quality (Estimar qualidade), incluindo as seguintes propriedades:

- AArea under the Precision-Recall curve (Área na curva de precisão-revocação) é um único número que estima o limite máximo da qualidade geral da transformação. Ele é independente da escolha feita para o parâmetro de precisão-revocação. Valores mais altos indicam que você tem uma compensação de precisão-revocação mais atrativa.
- Precision (Precisão) estima a frequência de acertos da transformação ao prever uma correspondência.
- Recall upper limit (Limite máximo de revocação) estima com que frequência a transformação prevê uma correspondência real.
- F1 estima a precisão da transformação entre 0 e 1, em que 1 é a maior precisão. Para obter mais informações, consulte F1 score na Wikipédia.
- A tabela Column importance (Importância da coluna) mostra os nomes das colunas e a pontuação de importância para cada uma. A importância da coluna ajuda você a entender como as colunas contribuem para o seu modelo, identificando quais colunas em seus registros estão sendo mais usadas para fazer a correspondência. Esses dados podem solicitar que você adicione ou altere seu conjunto de rótulos para aumentar ou diminuir a importância das colunas.

  A coluna Importance (Importância) fornece uma pontuação numérica para cada coluna, como um decimal não superior a 1,0.

Para obter informações sobre como entender as estimativas de qualidade com relação à qualidade verdadeira, consulte Estimativas de qualidade versus qualidade end-to-end (verdadeira).

Para obter mais informações sobre como ajustar a transformação, consulte Ajustar transformações de machine learning no AWS Glue.

Estimativas de qualidade versus qualidade end-to-end (verdadeira)

O AWS Glue estima a qualidade da transformação apresentando ao modelo de machine learning interno vários pares de registros para os quais você forneceu rótulos correspondentes, mas que o modelo ainda não tinha visto. Essas estimativas de qualidade são uma função da qualidade do modelo de machine learning (que é influenciado pelo número de registros que você rotula para "ensinar" a transformação). O end-to-end, ou recall verdadeiro (que não é calculado automaticamente pelo`ML transform`) também é influenciado pelo mecanismo de `ML transform` filtragem que propõe uma ampla variedade de combinações possíveis com o modelo aprendido por máquina.

É possível ajustar esse método de filtragem principalmente especificando o valor de ajuste como Menor precisão de custos. À medida que o valor de ajuste tende mais a favorecer a

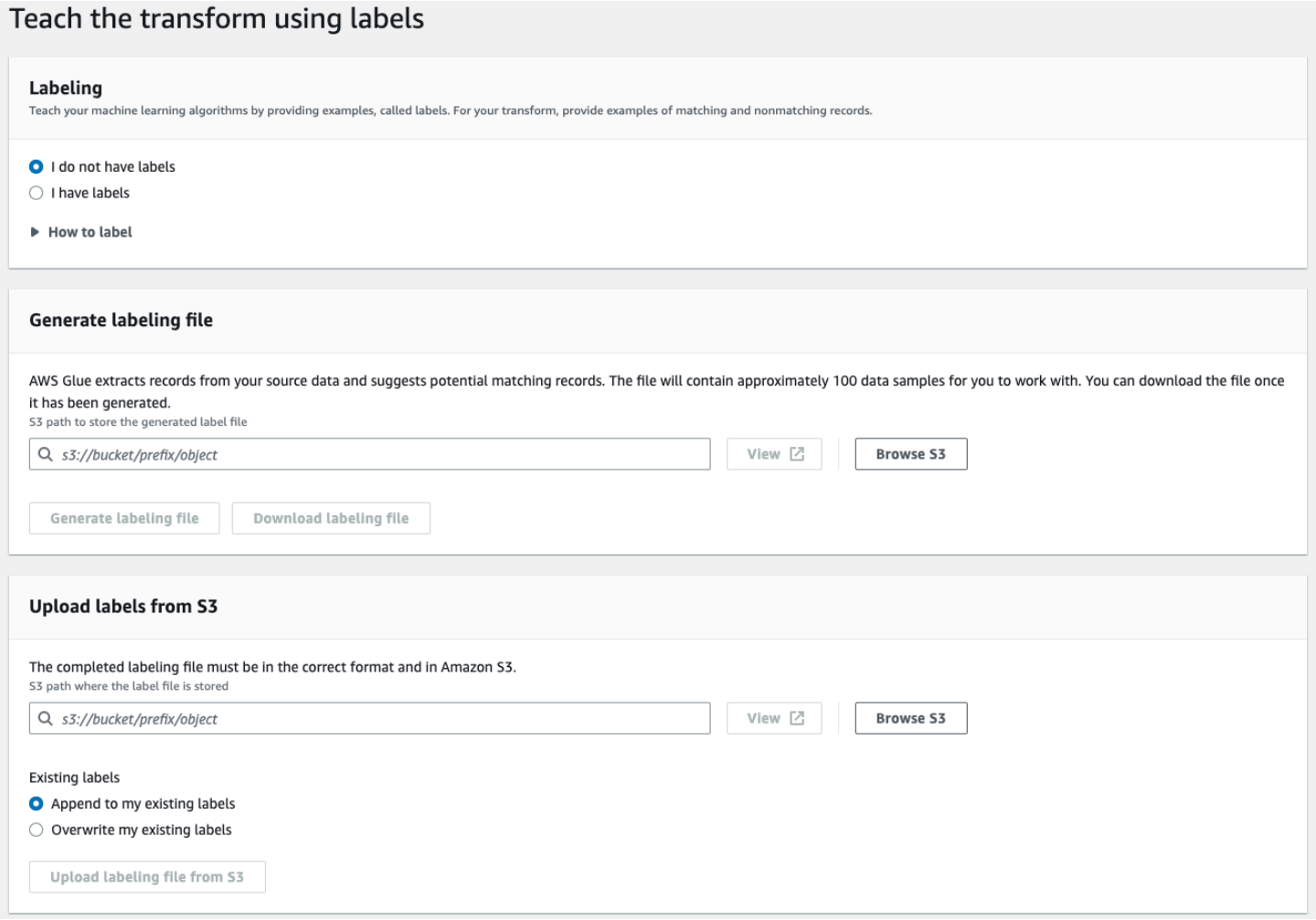
Precisão, o sistema faz uma busca mais detalhada e cara de pares de registros que podem ser correspondências. Mais pares de registros são inseridos em seu modelo aprendido por máquina, e sua recordação verdadeira se aproxima `ML transform` da end-to-end métrica de recall estimada. Como resultado, mudanças na end-to-end qualidade de suas correspondências como resultado de mudanças na relação custo/precisão de suas correspondências normalmente não serão refletidas na estimativa de qualidade.

Tags

As tags são rótulos que você pode atribuir a um AWS recurso. Cada tag consiste em uma chave e um valor opcional. As tags podem ser usadas para pesquisar e filtrar seu recurso ou rastrear seus AWS custos.

Ensine transformações usando rótulos

Você pode ensinar a transformação de ML usando rótulos (exemplos) escolhendo Ensinar a transformação na página de detalhes da transformação de ML. Ao ensinar seu algoritmo de machine learning fornecendo exemplos (denominados rótulos), você pode escolher usar rótulos existentes ou criar um arquivo de rótulo.



- Rótulagem: se você tiver rótulos escolha Eu tenho rótulos. Se você não tiver rótulos, ainda poderá continuar na próxima etapa gerando de um arquivo de rótulo.
- Gerar arquivo de rotulagem: o AWS Glue extrai os registros dos dados de origem e sugere possíveis registros correspondentes. Você escolhe o bucket do Amazon S3 para armazenar o arquivo de rótulo gerado. Escolha Gerar arquivo de rotulagem para iniciar o processo. Quando terminar, escolha Baixar arquivo de rotulagem. O arquivo baixado terá uma coluna para rótulos que você poderá preencher com os rótulos.
- Carregar rótulos do Amazon S3: escolha o arquivo de rotulo completo do bucket do Amazon S3 em que o arquivo de rótulo está armazenado. Depois, escolha entre anexar os rótulos aos rótulos existentes ou substituir os rótulos existentes. Escolha Carregar arquivo de rotulagem do Amazon S3.

Tutorial: como criar uma transformação de machine learning com o AWS Glue

Este tutorial orienta você sobre as ações para criar e gerenciar uma transformação de machine learning (ML) usando o AWS Glue. Antes de usar este tutorial, você deverá familiarizar-se com o uso do console do AWS Glue para adicionar crawlers a trabalhos e editar scripts. Você também deve se familiarizar com a localização e o download de arquivos no console do Amazon Simple Storage Service (Amazon S3).

Neste exemplo, você vai criar uma transformação `FindMatches` para encontrar registros correspondentes, ensiná-la como identificar registros correspondentes e não correspondentes e usá-la em um trabalho do AWS Glue. O trabalho do AWS Glue grava um novo arquivo do Amazon S3 com uma coluna adicional chamada `match_id`.

Os dados de origem usados por este tutorial é um arquivo chamado `dblp_acm_records.csv`. Esse arquivo é uma versão modificada de publicações acadêmicas (DBLP e ACM) disponíveis no Conjunto de dados de DBLP ACM original. O arquivo `dblp_acm_records.csv` é um arquivo de valores separados por vírgula (CSV) no formato UTF-8 sem marca de ordem de bytes (BOM).

Um segundo arquivo, `dblp_acm_labels.csv`, é um exemplo de arquivo de rotulagem que contém registros correspondentes e não correspondentes usados para ensinar a transformação como parte do tutorial.

Tópicos

Etapa 1: rastrear os dados de origem

Primeiro, rastreie o arquivo CSV de origem do Amazon S3 para criar uma tabela de metadados correspondente no Data Catalog.

> **Important**
>
> Para direcionar o crawler a criar uma tabela somente para o arquivo CSV, armazene os dados de origem do CSV em uma pasta do Amazon S3 separada de outros arquivos.

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Crawlers, Add crawler (Adicionar crawler).
3. Siga o assistente para criar e executar um crawler chamado `demo-crawl-dblp-acm` com a saída para o banco de dados `demo-db-dblp-acm`. Ao executar o assistente, crie o banco de dados, `demo-db-dblp-acm` se ele ainda não existir. Escolha um caminho de inclusão do Amazon S3 para os dados de exemplo na região da AWS atual. Por exemplo, para `us-east-1`, o caminho de inclusão do Amazon S3 para o arquivo de origem será `s3://ml-transforms-public-datasets-us-east-1/dblp-acm/records/dblp_acm_records.csv`.

   Se for bem-sucedido, o crawler criará a tabela `dblp_acm_records_csv` com as seguintes colunas: id, título, autores, local, ano e origem.

Etapa 2: adicionar uma transformação de machine learning

Em seguida, adicione uma transformação de machine learning que se baseia no esquema da tabela de fonte de dados criada pelo crawler chamado `demo-crawl-dblp-acm`.

1. No console do AWS Glue, no painel de navegação em Integração de dados e ETL, escolha Ferramentas de classificação de dados > Correspondência de registros e depois Adicionar transformação. Siga o assistente para criar uma transformação `Find matches` com as propriedades a seguir.

   a. Em Transform name (Nome da transformação), insira **`demo-xform-dblp-acm`**. Este é o nome da transformação que é usada para localizar correspondências nos dados de origem.

   b. Em IAM role (Função do IAM), escolha uma função do IAM que tenha permissão para os dados de origem do Amazon S3, o arquivo de rotulagem e as operações da API do AWS Glue. Para obter mais informações, consulte Criar uma função do IAM para o AWS Glue no Guia do desenvolvedor do AWS Glue.

   c. Em Data source (Fonte de dados), escolha a tabela chamada dblp_acm_records_csv no banco de dados demo-db-dblp-acm.

d. Em Primary key (Chave primária), escolha a coluna de chave primária da tabela id.

2. No assistente, escolha Finish (Concluir) e retorne à lista ML transforms (Transformações de ML).

Etapa 3: ensinar sua transformação de machine learning

Em seguida, você ensina sua transformação de machine learning usando o arquivo de rotulagem de exemplo do tutorial.

Não é possível usar uma transformação de linguagem de máquina em um trabalho de extração, transformação e carga (ETL) até que o status dela seja Ready for use (Pronta para uso). Para que sua transformação fique pronta, você deve ensiná-la como identificar registros correspondentes e não correspondentes fornecendo exemplos desses registros. Para ensinar a transformação, você pode Generate a label file (Gerar um arquivo de rótulos), adicionar rótulos e Upload label file (Fazer upload do arquivo de rótulos). Neste tutorial, você poderá usar o arquivo de rotulagem de exemplo chamado `dblp_acm_labels.csv`. Para obter mais informações sobre o processo de rotulagem, consulte Rótulo.

1. No console do AWS Glue, no painel de navegação, escolha Correspondência de registros.
2. Escolha a transformação `demo-xform-dblp-acm` e escolha Action (Ação), Teach (Ensinar). Siga o assistente para ensinar a transformação `Find matches`.
3. Na página de propriedades da transformação, escolha I have labels (Tenho rótulos). Escolha um caminho do Amazon S3 para o arquivo de rotulagem de exemplo na região da AWS atual. Por exemplo, para `us-east-1`, faça upload do arquivo de rotulagem fornecido no caminho do Amazon S3 `s3://ml-transforms-public-datasets-us-east-1/dblp-acm/labels/dblp_acm_labels.csv` com a opção de overwrite (substituir) os rótulos existentes. O arquivo de rotulagem deve estar localizado no Amazon S3 na mesma região que o console do AWS Glue.

   Quando você faz upload de um arquivo de rotulagem, uma tarefa é iniciada no AWS Glue para adicionar ou substituir os rótulos usados para ensinar a transformação como processar a fonte de dados.
4. Na página final do assistente, escolha Finish (Concluir) e retorne à lista de ML transforms (Transformações de ML).

Etapa 4: estimar a qualidade de sua transformação de machine learning

É possível estimar a qualidade de sua transformação de machine learning. A qualidade depende da quantidade de rotulagem que você tiver feito. Para obter mais informações sobre como estimar a qualidade, consulte Estimar qualidade.

1. No console do AWS Glue, no painel de navegação em Integração de dados e ETL, escolha Ferramentas de classificação de dados > Correspondência de registros.
2. Escolha a transformação `demo-xform-dblp-acm` e escolha a guia Estimate quality (Estimar qualidade). Essa guia exibe as estimativas de qualidade atuais, se disponível, para a transformação.
3. Escolha Estimate quality (Estimar qualidade) para iniciar uma tarefa para estimar a qualidade da transformação. A precisão da estimativa da qualidade é baseada na rotulagem dos dados de origem.
4. Navegue até a guia History (Histórico). Nesse painel, estão listadas as execuções de tarefa para a transformação, incluindo a tarefa Estimating quality (Estimar qualidade). Para obter mais detalhes sobre a execução, escolha Logs. Verifique se o status da execução é Succeeded (Bem-sucedido) após a conclusão.

Etapa 5: adicionar e executar um trabalho com sua transformação de machine learning

Nesta etapa, você usará a transformação de machine learning para adicionar e executar um trabalho no AWS Glue. Quando a transformação `demo-xform-dblp-acm` estiver Ready for use (Pronta para uso), você poderá usá-la em um trabalho de ETL.

1. No console do AWS Glue, no painel de navegação, escolha Jobs (Trabalhos).
2. Escolha Add job (Adicionar trabalho), e siga as etapas do assistente para criar um trabalho de ETL do Spark com um script gerado. Escolha os seguintes valores de propriedade para sua transformação:

   a. Em Name (Nome), escolha o trabalho de exemplo neste tutorial, demo-etl-dblp-acm.

   b. Em IAM role (Função do IAM), escolha uma função do IAM com permissão para os dados de origem do Amazon S3, o arquivo de rotulagem e as operações da API do AWS Glue. Para obter mais informações, consulte Criar uma função do IAM para o AWS Glue no Guia do desenvolvedor do AWS Glue.

   c. Em ETL language (Linguagem de ETL), escolha Scala. Esta será a linguagem de programação no script de ETL.

d. Em Script file name (Nome do arquivo de script), escolha demo-etl-dblp-acm. Este é o nome do arquivo do script Scala (igual ao nome do trabalho).

e. Em Data source (Fonte de dados), escolha dblp_acm_records_csv. A fonte de dados que você escolher deve corresponder ao esquema da fonte de dados da transformação de machine learning.

f. Em Transform type (Tipo de transformação), escolha Find matching records (Encontrar registros correspondentes) para criar um trabalho usando uma transformação de machine learning.

g. Desmarque Remove duplicate records (Remover registros duplicados). Não remova os registros duplicados porque os registros de saída de gravados têm um campo extra `match_id` adicionado.

h. Em Transform (Transformação), escolha a transformação de machine learning demo-xform-dblp-acm usada pelo trabalho.

i. Em Create tables in your data target (Criar tabelas em seu destino de dados), escolha criar tabelas com as seguintes propriedades:

- Data store type (Tipo de armazenamento de dados): **`Amazon S3`**
- Format (Formato): **`CSV`**
- Compression type (Tipo de compactação): **`None`**
- Target path (Caminho de destino): o caminho do Amazon S3 onde a saída do trabalho é gravada (no console da região da AWS atual)

3. Escolha Save job and edit script (Salvar trabalho e editar script) para exibir a página do editor de script.

4. Edite o script para adicionar uma instrução para fazer com que a saída do trabalho para o Target path (Caminho de destino) seja gravado em um único arquivo de partição. Adicione essa instrução imediatamente após a instrução que executa a transformação `FindMatches`. A instrução é semelhante ao seguinte.

```
val single_partition = findmatches1.repartition(1)
```

Você deve modificar a instrução `.writeDynamicFrame(findmatches1)` para gravar a saída como `.writeDynamicFrame(single_partion)`.

5. Depois de editar o script, escolha Save (Salvar). O script modificado é similar ao código a seguir, mas personalizado para seu ambiente.

```
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()
  }
}
```

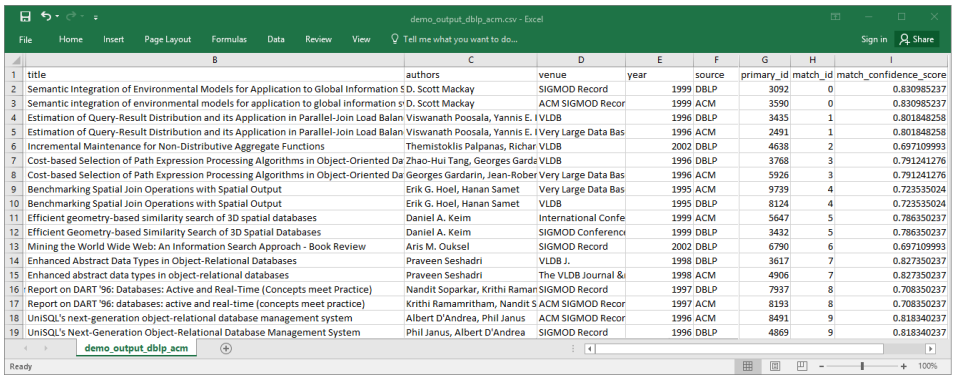
6. Escolha Run job (Executar trabalho) para iniciar a execução do trabalho. Verifique o status do trabalho na lista de trabalhos. Quando o trabalho terminar, em ML transform (Transformação de ML), guia History (Histórico), haverá uma nova linha adicionada Run ID (ID de execução) do tipo ETL job (Trabalho de ETL).
7. Navegue até Jobs (Trabalhos), guia History (Histórico). Nesse painel, estão listadas as execuções de trabalho. Para obter mais detalhes sobre a execução, escolha Logs. Verifique se o status da execução é Succeeded (Bem-sucedido) após a conclusão.

Etapa 6: verificar os dados de saída do Amazon S3

Nesta etapa, verifique a saída da execução do trabalho no bucket do Amazon S3 que você escolheu quando adicionou o trabalho. Você pode fazer download do arquivo de saída em sua máquina local e verificar se os registros correspondentes foram identificados.

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Faça download do arquivo de saída de destino do trabalho `demo-etl-dblp-acm`. Abra o arquivo em um aplicativo de planilha (talvez seja necessário adicionar uma extensão de arquivo `.csv` para que o arquivo abra corretamente).

   A imagem a seguir mostra um trecho da saída no 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

O arquivo da fonte de dados e o arquivo do destino têm 4.911 registros. No entanto, a transformação `Find matches` adiciona outra coluna chamada `match_id` para identificar registros correspondentes na saída. As linhas com o mesmo `match_id` são consideradas registros correspondentes. O `match_confidence_score` é um número entre 0 e 1 que fornece uma estimativa da qualidade das correspondências encontradas por `Find matches`.

3. Classifique o arquivo de saída por `match_id` para ver facilmente quais registros são correspondentes. Compare os valores nas outras colunas para ver se você concorda com os resultados da transformação `Find matches`. Se você não concordar, poderá continuar a ensinar a transformação adicionando mais rótulos.

   Também é possível classificar o arquivo por outro campo, como `title`, para ver se os registros com títulos semelhantes têm o mesmo `match_id`.

Localizar correspondências incrementais

O recurso Find matches permite identificar registros duplicados ou correspondentes no seu conjunto de dados, mesmo quando os registros não tenham um identificador exclusivo comum e quando não houver campos com uma correspondência exata. A versão inicial da transformação Find matches identificava registros correspondentes em um único conjunto de dados. Quando você adiciona novos dados ao conjunto de dados, você precisava mesclá-los com o conjunto de dados limpo existente e executar novamente a correspondência com o conjunto de dados mesclado completo.

O recurso de correspondência incremental facilita a correspondência com registros incrementais em relação a conjuntos de dados correspondentes existentes. Vamos supor que você queira combinar dados de clientes potenciais com conjuntos de dados de clientes existentes. O recurso de correspondência incremental oferece a flexibilidade de combinar centenas de milhares de novos clientes potenciais com um banco de dados existente de clientes potenciais e clientes, mesclando os resultados em um único banco de dados ou tabela. Ao fazer a correspondência somente entre os conjuntos de dados novos e existentes, a otimização da localização de correspondências incrementais reduz o tempo de computação, o que também reduz o custo.

O uso de correspondência incremental é semelhante a Find matches conforme descrito em Tutorial: como criar uma transformação de machine learning com o AWS Glue. Este tópico identifica apenas as diferenças com correspondência incremental.

Para obter mais informações, consulte a postagem do blog em Correspondências de dados incrementais.

Execução de um trabalho de correspondência incremental

Para o procedimento a seguir, suponha o seguinte:

- Você rastreou o conjunto de dados existente até a tabela first_records. O conjunto de dados first_records deve ser um conjunto de dados correspondente ou a saída do trabalho correspondente.
- Você criou e treinou uma transformação de Find matches com o AWS Glue versão 2.0. Essa é a única versão do AWS Glue com suporte a correspondências incrementais.
- A linguagem de ETL é Scala. Observe que também há suporte para Python.
- O modelo já gerado é chamado de `demo-xform`.

1. Rastreie o conjunto de dados incremental até a tabela second_records.
2. No console do AWS Glue, no painel de navegação, escolha Jobs (Trabalhos).
3. Escolha Add job (Adicionar trabalho), e siga as etapas do assistente para criar um trabalho de ETL do Spark com um script gerado. Escolha os seguintes valores de propriedade para sua transformação:

   a. Em Name (Nome), escolhademo-etl.

   b. Em IAM role (Função do IAM), escolha uma função do IAM com permissão para os dados de fonte do Amazon S3, o arquivo de rotulagem e as operações de API do AWS Glue.

   c. Em ETL language (Linguagem de ETL), escolha Scala.

   d. Em Script file name (Nome do arquivo de script), escolha demo-etl. Esse é o nome do arquivo do script Scala.

   e. Em Data source (Fonte de dados), escolha first_records. A fonte de dados que você escolher deve corresponder ao esquema da fonte de dados da transformação de machine learning.

   f. Em Transform type (Tipo de transformação), escolha Find matching records (Encontrar registros correspondentes) para criar um trabalho usando uma transformação de machine learning.

   g. Selecione a opção de correspondência incremental e para Data Source (Fonte de dados), selecione a tabela chamada second_records.

   h. Em Transform (Transformação), escolha demo-xform, a transformação de machine learning usada pelo trabalho.

   i. Escolha Create tables in your data target (Criar tabelas em seu destino de dados) ou Use tables in the data catalog and update your data target (Usar tabelas no catálogo de dados e atualizar seu destino de dados).
4. Escolha Save job and edit script (Salvar trabalho e editar script) para exibir a página do editor de script.
5. Escolha Run job (Executar trabalho) para iniciar a execução do trabalho.

Usar FindMatches em um trabalho visual

Para usar a transformação FindMatches no AWS Glue Studio, você pode usar o nó Transformação personalizada que invoca a API FindMatches. Para obter mais informações sobre como usar uma transformação personalizada, consulte Criar uma transformação personalizada

> **Note**
>
> Atualmente, a API FindMatches só funciona com o `Glue 2.0`. Para executar um trabalho com a transformação personalizada que invoca a API FindMatches, verifique se a versao do AWS Glue é **`Glue 2.0`** na guia Detalhes do trabalho. Se a versão do AWS Glue não for `Glue 2.0`, o trabalho falhará no runtime com a seguinte mensagem de erro: “não é possível importar o nome 'FindMatches' de 'awsglueml.transforms'”.

Pré-requisitos

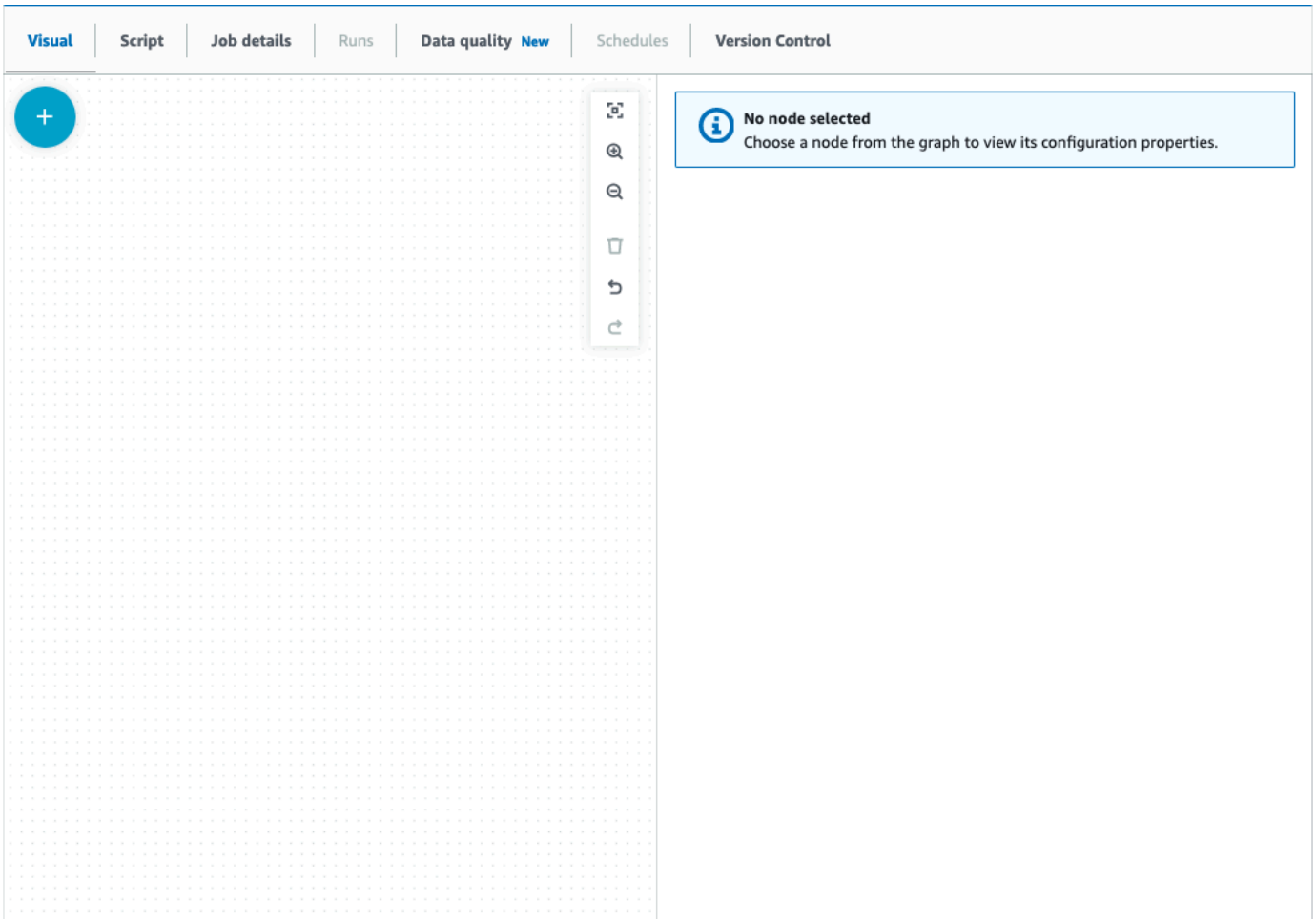
- Para usar a transformação Find Matches, abra o console do AWS Glue Studio no https://console.aws.amazon.com/gluestudio/.
- Cria uma transformação de machine learning. Quando criada, um transformId é gerado. Você precisará desse ID nas etapas abaixo. Para obter mais informações sobre como criar uma transformação de machine learning, consulte Adicionar e editar transformações de machine learning.

Adicionar uma transformação FindMatches

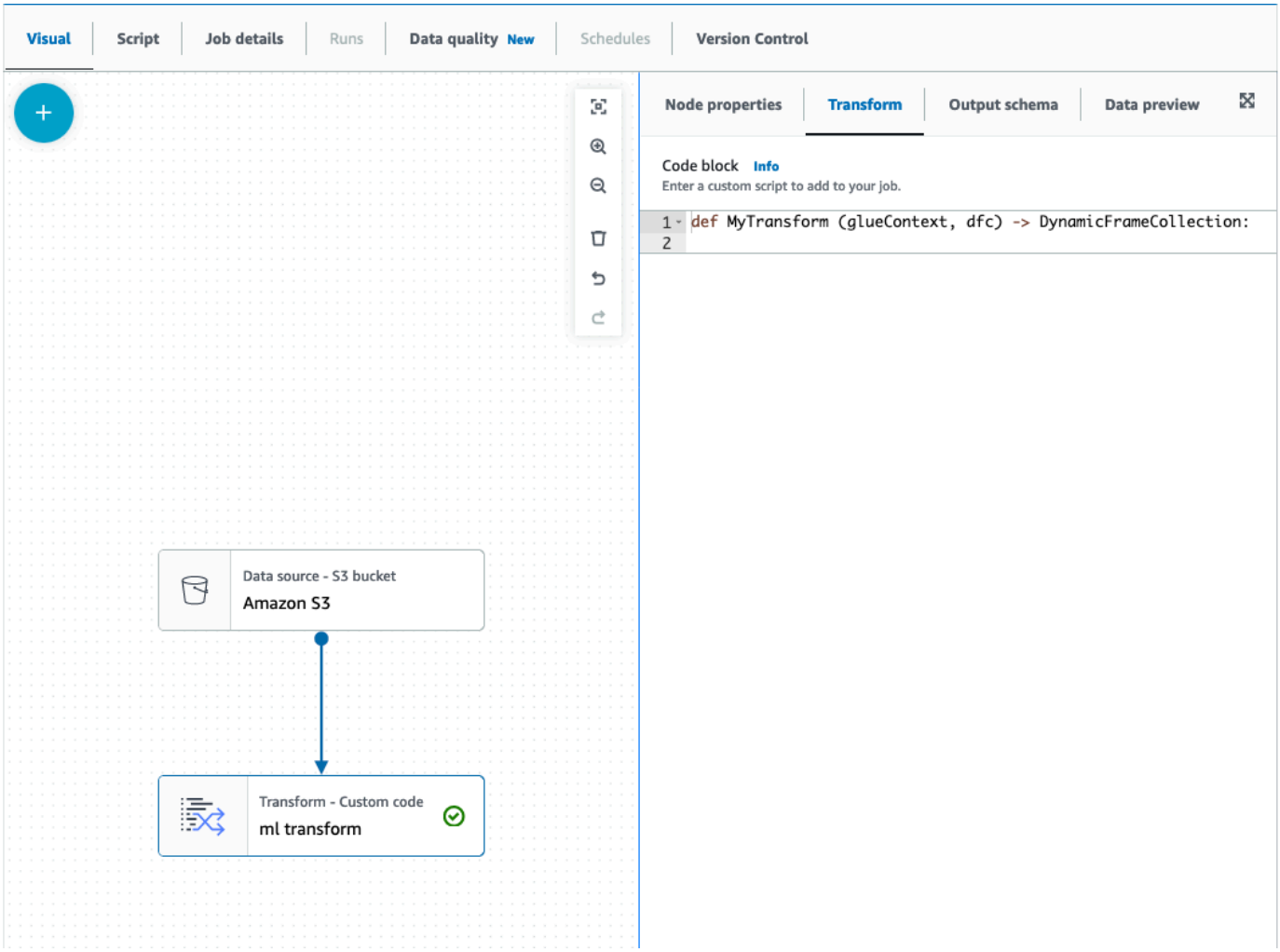
Para adicionar uma transformação FindMatches:

1. No editor de trabalho do AWS Glue Studio, abra o painel Recurso clicando no símbolo de cruz no canto superior esquerdo do gráfico visual do trabalho e escolha uma fonte de dados

selecionando a guia Dados. Essa é a fonte de dados na qual você deseja verificar se há correspondências.



2. Escolha o nó da fonte de dados e abra o painel Recurso clicando no símbolo de cruz no canto superior esquerdo do gráfico visual do trabalho e procure “transformação personalizada”. Escolha o nó Transformação personalizada para adicioná-lo ao gráfico. A transformação personalizada é vinculada ao nó da fonte de dados. Caso contrário, você pode clicar no nó Transformação personalizada e escolher a guia Propriedades do nó e, em Superiores do nó, escolher a fonte de dados.
3. Clique no nó Transformação personalizada no gráfico visual, depois escolha a guia Propriedades do nó e dê um nome para a transformação personalizada. É recomendável renomear a transformação para que o nome da transformação seja facilmente identificável no gráfico visual.
4. Escolha a guia Transformar, na qual você pode editar o bloco de código. É aqui que o código para invocar a API FindMatches pode ser adicionado.



O bloco de código contém código já preenchido para você começar. Substitua o código já preenchido pelo modelo abaixo. O modelo tem um espaço reservado para o transformId, o qual você pode fornecer.

```
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. Clique no nó Transformação personalizada no gráfico visual, abra o painel Recurso clicando no símbolo de cruz no canto superior esquerdo do gráfico visual do trabalho e procure “transformação personalizada”. Não há necessidade de alterar a seleção padrão, pois há somente um DynamicFrame no conjunto.

6. Você pode continuar adicionando transformações ou armazenar o resultado, que agora está enriquecido com as colunas adicionais de descoberta de correspondências. Se você quiser referenciar essas novas colunas em transformações posteriores, precisará adicioná-las ao esquema de saída da transformação. A maneira mais fácil de fazer isso é escolher a guia Visualização de dados e, na guia Esquema, escolher “Usar esquema de visualização de dados”.
7. Para personalizar findMatches, você pode adicionar outros parâmetros para passar para o método 'aplicar'. Consulte Classe FindMatches.

Adicionar uma transformação incremental FindMatches

No caso de correspondências incrementais, o processo é o mesmo que Adicionar uma transformação FindMatches com as seguintes diferenças:

- Em vez de um nó superior para a transformação personalizada, você precisará de dois nós superiores.
- O primeiro nó superior deve ser o conjunto de dados.
- O segundo nó superior deve ser o conjunto de dados incrementais.

  Substitua o `transformId` pelo seu `transformId` no bloco de código do modelo:

```
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 obter parâmetros opcionais, consulte a classe FindIncrementalMatches.

Migrar programas do Apache Spark para o AWS Glue

O Apache Spark é uma plataforma de código aberto para workloads de computação distribuída executadas em grandes conjuntos de dados. O AWS Glue utiliza os recursos do Spark para fornecer uma experiência otimizada para ETL. Você pode migrar programas Spark para o AWS Glue para

utilizar nossos recursos. O AWS Glue fornece os mesmos aprimoramentos de performance que você esperaria do Apache Spark no Amazon EMR.

Executar o código Spark

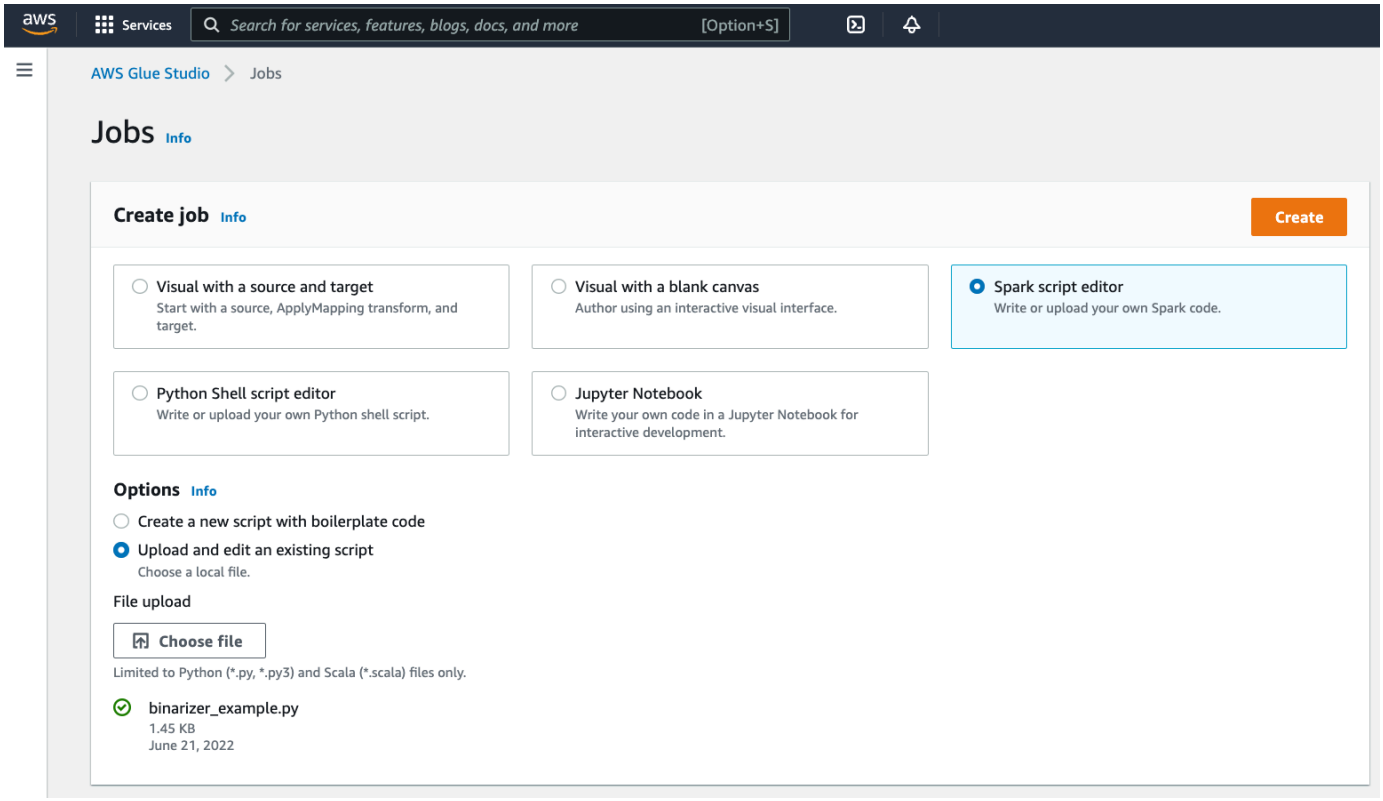
O código nativo do Spark pode ser executado em um ambiente AWS Glue pronto para uso. Os scripts muitas vezes são desenvolvidos alterando iterativamente um trecho de código, um fluxo de trabalho adequado para uma sessão interativa. No entanto, convém executar o código existente em um trabalho do AWS Glue, que permite agendar e obter consistentemente logs e métricas para cada execução de script. Você pode carregar e editar um script existente pelo console.

1. Adquira a origem de seu script. Para este exemplo, você usará um script de exemplo do repositório do Apache Spark. Exemplo de binarizador
2. No console do AWS Glue, expanda o painel de navegação do lado esquerdo e selecione ETL > Jobs (Trabalhos)

   No painel Create job (Criar trabalho), selecione Spark script editor (Editor de scripts Spark). Será exibida a seção Options (Opções). Em Options (Opções), clique em Upload and edit an existing script (Carregar e editar um script existente).

   Será exibida a seção File upload (Carregamento de arquivo). Em File upload (Carregamento de arquivo), clique em Choose file (Escolher arquivo). Será exibido seletor de arquivos do sistema. Navegue até o local em que você salvou `binarizer_example.py`, selecione-o e confirme a seleção.

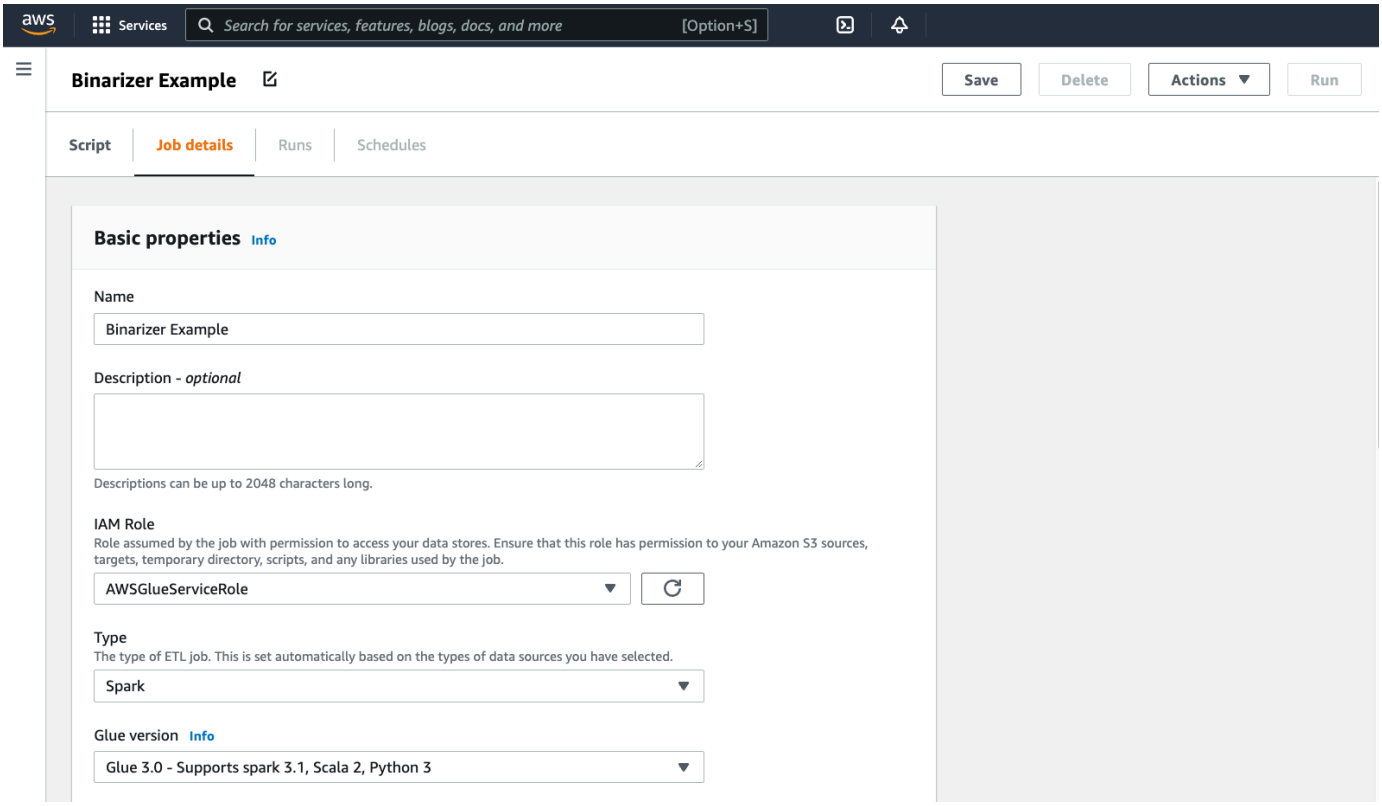
   Será exibido um botão Create (Criar) no cabeçalho do painel Create job (Criar trabalho). Clique.



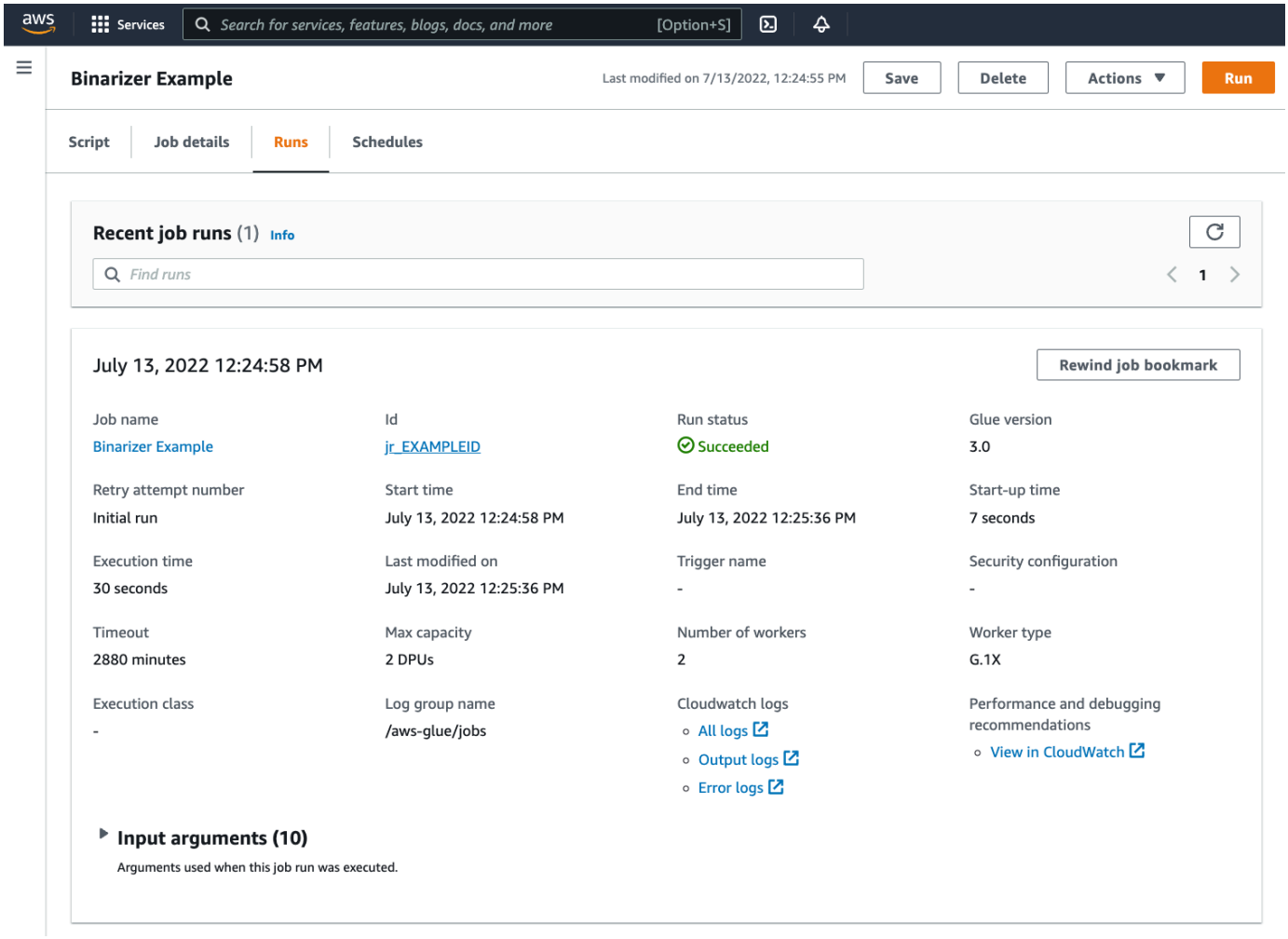
3. Seu navegador conduzirá até o editor de scripts. No cabeçalho, clique na guia Job details (Detalhes do trabalho). Defina o nome e o perfil do IAM. Para obter orientação sobre perfis do IAM do AWS Glue, consultethe section called “Configurar permissões do IAM”.

   Opcionalmente: defina Requested number of workers (Número solicitado de operadores) para 2 e Number of retries (Número de repetições) para 1. Essas opções são proveitosas para executar trabalhos de produção, mas recusá-las simplificará sua experiência ao testar um recurso.

   Na barra de título, clique em Save (Salvar) e depois em Run (Executar)

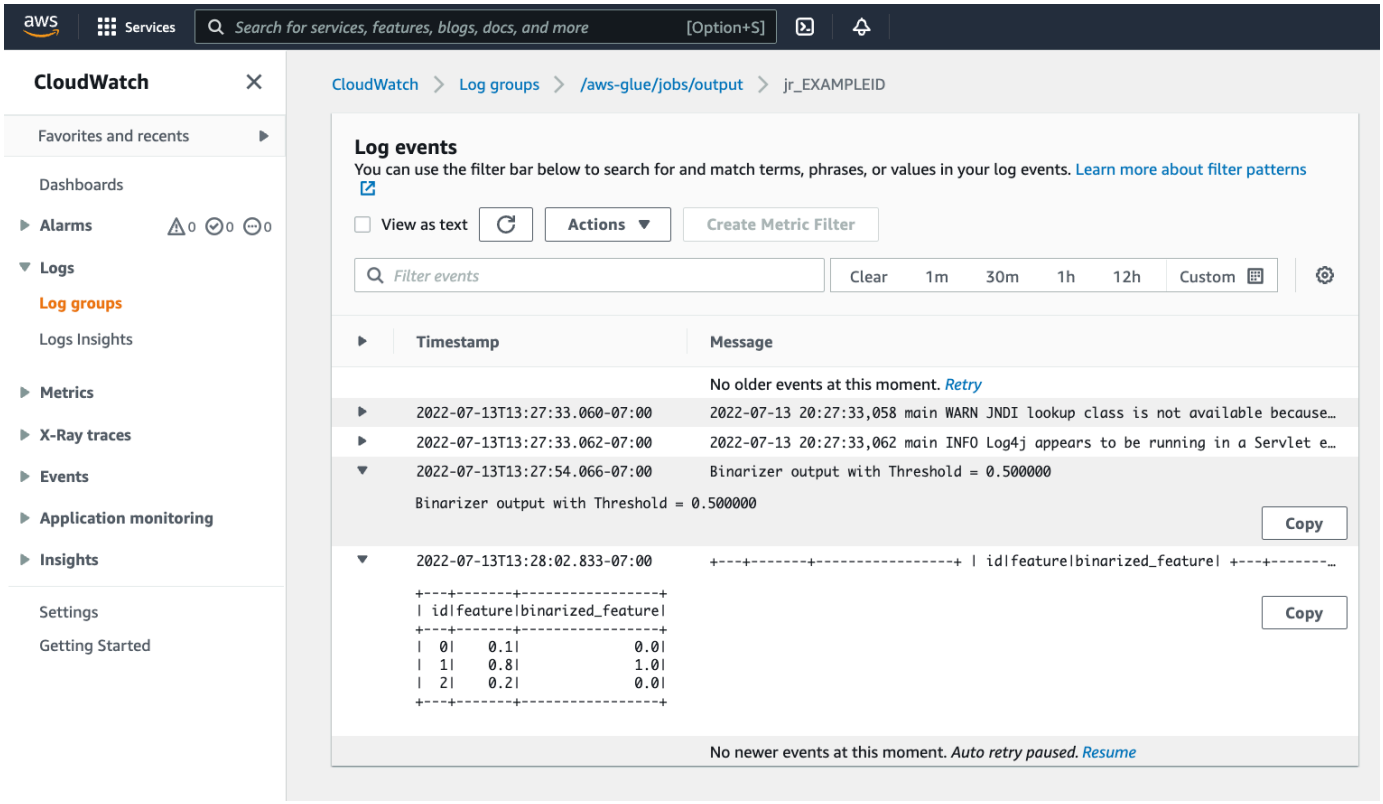


4. Navegue até a guia Runs (Execuções). Você verá um painel correspondente à execução do trabalho. Aguarde alguns instantes, e a página deverá atualizar automaticamente para exibir Succeeded (Êxito) em Run status (Status da execução).



5. Convém examinar sua saída para confirmar que o script Spark foi executado conforme o esperado. Esse script de amostra do Apache Spark deve gravar uma cadeia de caracteres no fluxo de saída. Você pode descobrir isso navegando até Output logs (Logs de saída) em Logs no painel para a execução bem-sucedida do trabalho. Observe o ID de execução do trabalho, que é um ID gerado no rótulo Id começando com `jr_`.

   Isso abrirá o console do CloudWatch, definido para visualizar o conteúdo do grupo de logs padrão `/aws-glue/jobs/output` do AWS Glue, filtrado para o conteúdo dos fluxos de logs para o ID de execução do trabalho. Cada operador terá gerado um fluxo de logs, mostrado como linhas em Log streams (Fluxos de log). Um operador deverá ter executado o código solicitado. Você precisará abrir todos os fluxos de logs para identificar o operador correto. Depois de encontrar o operador certo, você deverá ver a saída do script, como é exibido na imagem a seguir:



Procedimentos comuns necessários para migrar programas Spark

Avalie a compatibilidade com a versão do Spark

As versões do AWS Glue determinam a versão do Apache Spark e Python que estão disponíveis para o trabalho do AWS Glue. Você encontra nossas versões do AWS Glue e a compatibilidade delas em the section called “Versões do AWS Glue”. Pode ser necessário atualizar seu programa Spark para que seja compatível com uma versão mais recente do Spark para acessar determinados recursos do AWS Glue.

Incluir bibliotecas de terceiros

Muitos programas Spark existentes terão dependências, tanto em artefatos privados como públicos. O AWS Glue é compatível com dependências de estilo JAR para trabalhos Scala, bem como dependências Wheel e Python puro para trabalhos Python.

Python: para obter informações sobre dependências do Python, consulte the section called “Bibliotecas Python”

As dependências comuns do Python são fornecidas no ambiente do AWS Glue, inclusive a popular biblioteca Pandas. As dependências estão incluídas no AWS Glue versão 2.0+. Para obter mais

informações sobre os módulos fornecidos, consulte the section called “Módulos do Python já fornecidos no AWS Glue”. Se você precisar fornecer a um trabalho uma versão diferente de uma dependência incluída por padrão, você pode usar `--additional-python-modules`. Para obter mais informações sobre argumentos de trabalho, consulte the section called “Parâmetros de trabalho”.

Você pode fornecer dependências adicionais do Python com o argumento de trabalho `--extra-py-files`. Se você estiver migrando um trabalho de um programa Spark, esse parâmetro é uma boa opção, pois é funcionalmente equivalente ao sinalizador `--py-files` no PySpark e está sujeito às mesmas limitações. Para obter mais informações sobre o parâmetro `--extra-py-files`, consulte the section called “Incluindo arquivos Python com recursos nativos PySpark ”

Para novos trabalhos, é possível gerenciar dependências do Python com o argumento de trabalho `--additional-python-modules`. Usar esse argumento permite uma experiência mais completa de gerenciamento de dependências. Esse parâmetro é compatível com dependências do estilo Wheel, inclusive aquelas com vinculações de código nativas compatíveis com o Amazon Linux 2.

Scala

Você pode fornecer dependências adicionais do Scala com o argumento de trabalho `--extra-jars`. As dependências devem ser hospedadas no Amazon S3, e o valor do argumento deve ser uma lista delimitada por vírgulas de caminhos do Amazon S3 sem espaços. Talvez seja mais fácil gerenciar sua configuração reagrupando suas dependências antes de hospedá-las e configurá-las. AWS Glue As dependências JAR contêm bytecode Java, que pode ser gerado de qualquer linguagem JVM. É possível usar outras linguagens JVM, como Java, para escrever dependências personalizadas.

Gerenciar credenciais da fonte de dados

Os programas Spark existentes podem vir com configurações complexas ou personalizadas para extrair dados das fontes de dados. Fluxos de autenticação de fonte de dados comuns são compatíveis com conexões do AWS Glue. Para obter mais informações sobre conexões do AWS Glue, consulte Conectar a dados.

As conexões do AWS Glue permitem a conexão do trabalho a uma variedade de tipos de armazenamentos de dados, principalmente de duas maneiras: por meio de chamadas de método para nossas bibliotecas e definindo Additional network connection (Conexão de rede adicional) no console do AWS. Também é possível chamar o SDK da AWS de dentro do trabalho para recuperar informações de uma conexão.

Chamadas de método: as conexões do AWS Glue são totalmente integradas ao AWS Glue Data Catalog, um serviço que permite selecionar informações sobre seus conjuntos de dados, e os métodos disponíveis para interagir com conexões do AWS Glue refletem isso. Se houver uma configuração de autenticação existente que você gostaria de reutilizar, para conexões JDBC, é possível acessar a configuração de conexão do AWS Glue pelo método `extract_jdbc_conf` em `GlueContext`. Para obter mais informações, consulte the section called “extract_jdbc_conf”

Configuração do console: o uso de trabalhos do AWS Glue associados a conexões do AWS Glue para configurar conexões com sub-redes da Amazon VPC. Se você gerencia diretamente seus materiais de segurança, talvez seja necessário fornecer um tipo `NETWORK` para Additional network connection (Conexão de rede adicional) no console do AWS para configurar o roteamento. Para obter mais informações sobre a conexão de APIs do AWS Glue, consulte the section called “Conexões”

Se seus programas Spark tiverem um fluxo de autenticação personalizado ou incomum, talvez seja necessário gerenciar seus materiais de segurança na prática. Se as conexões do AWS Glue não parecerem uma boa opção, você poderá hospedar seguramente materiais de segurança no Secrets Manager e acessá-los pelo boto3 ou pelo SDK da AWS, que são fornecidos no trabalho.

Configurar o Apache Spark

As migrações complexas geralmente alteram a configuração do Spark para acomodar suas workloads. As versões modernas do Apache Spark permitem que a configuração do runtime seja definida com o `SparkSession`. AWS Glue Trabalhos 3.0+ recebem `SparkSession`, que pode ser modificado para definir a configuração do runtime. Configuração do Apache Spark. Ajustar o Spark é complexo, e o AWS Glue não garante suporte para definir todas as configurações do Spark. Se sua migração necessitar de uma configuração substancial no nível do Spark, entre em contato com o suporte.

Definir uma configuração personalizada

Os programas Spark migrados podem ser projetados de modo a ter configurações personalizadas. O AWS Glue permite que a configuração seja definida no nível do trabalho e da execução do trabalho, por meio dos argumentos do trabalho. Para obter mais informações sobre argumentos de trabalho, consulte the section called “Parâmetros de trabalho”. É possível acessar argumentos de trabalho dentro do contexto de um trabalho por meio de nossas bibliotecas. O AWS Glue fornece uma função de utilitário que proporciona uma visão consistente entre os argumentos definidos no trabalho e os argumentos definidos na execução do trabalho. Consulte the section called “getResolvedOptions” em Python e the section called “GlueArgParser” em Scala.

Migrar código Java

Conforme explicado em the section called “Bibliotecas de terceiros”, suas dependências podem conter classes geradas por linguagens JVM, como Java ou Scala. Suas dependências podem incluir um método `main`. Você pode usar um método `main` em uma dependência como o ponto de entrada para um trabalho Scala do AWS Glue. Isso permite escrever o método `main` em Java ou reutilizar um método `main` empacotado de acordo com seus próprios padrões de biblioteca.

Para usar um método `main` de uma dependência, execute este procedimento: limpe o conteúdo do painel de edição fornecendo o objeto `GlueApp` padrão. Forneça o nome totalmente qualificado de uma classe em uma dependência como argumento de trabalho com a chave `--class`. Então, você deverá ser capaz de acionar uma execução de trabalho.

Não é possível configurar a ordem ou a estrutura dos argumentos. O AWS Glue passa para o método `main`. Se o código existente precisar ler a configuração definida no AWS Glue, isso provavelmente causará incompatibilidade com o código anterior. Se você usar `getResolvedOptions`, também não terá um bom lugar para chamar esse método. Considere invocar sua dependência diretamente de um método principal gerado pelo AWS Glue. O script de ETL do AWS Glue a seguir é um exemplo disso.

```
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()
  }
}
```

Trabalhar com trabalhos do Ray no AWS Glue

Esta seção fornece informações sobre o uso do AWS Glue para Ray. Para obter mais informações sobre escrever scripts do AWS Glue para Ray, consulte a seção the section called “AWS Glue para Ray”.

Tópicos

Conceitos básicos do AWS Glue para Ray

Para trabalhar com o AWS Glue para Ray, você usa os mesmos trabalhos e sessões interativas do AWS Glue que usaria com o AWS Glue para Spark. Os trabalhos do AWS Glue foram criados para executar o mesmo script em uma cadência recorrente, enquanto as sessões interativas foram criadas para permitir que você execute trechos de código sequencialmente nos mesmos recursos provisionados.

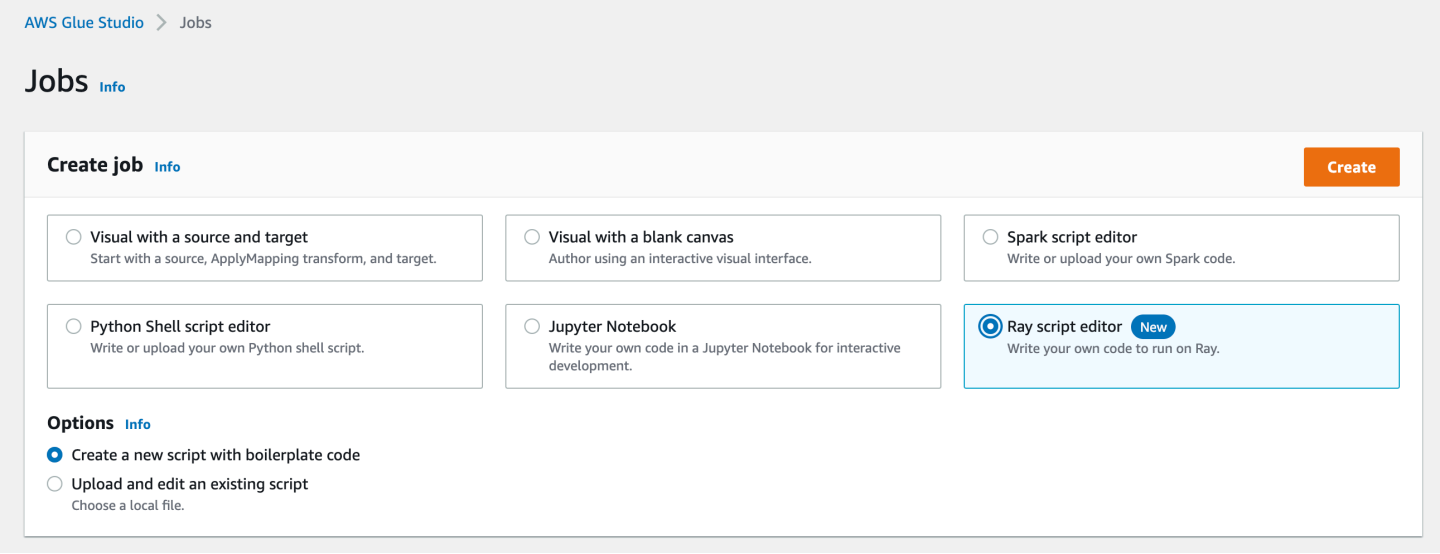
O AWS Glue ETL e o Ray são diferentes internamente, portanto, em seu script, você terá acesso a diferentes ferramentas, atributos e configurações. Como uma nova estrutura de computação gerenciada pelo AWS Glue, o Ray tem uma arquitetura diferente e usa um vocabulário diferente para descrever o que faz. Para obter mais informações, consulte Architecture Whitepapers (Documentos técnicos de arquitetura) na documentação do Ray.

> **Note**
>
> O AWS Glue for Ray está disponível em Leste dos EUA (Norte da Virgínia), Leste dos EUA (Ohio), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio) e Europa (Irlanda).

Trabalhos do Ray no console do AWS Glue Studio

Na página Trabalhos no console do AWS Glue Studio, você pode selecionar uma nova opção ao criar um trabalho no AWS Glue Studio - editor de script do Ray. Escolha essa opção para criar um

trabalho do Ray no console. Para obter mais informações sobre trabalhos e sobre como eles são usados, consulte Criar trabalhos ETL visuais com o AWS Glue Studio.



Os trabalhos do Ray na AWS CLI e SDK

Os trabalhos do Ray na AWS CLI usam as mesmas ações e parâmetros do SDK dos outros trabalhos. O AWS Glue para Ray introduz novos valores para determinados parâmetros. Para obter mais informações sobre a API de trabalhos, consulte the section called “Tarefas”.

Ambientes de runtime do Ray compatíveis

Em trabalhos do Spark, o `GlueVersion` determina as versões do Apache Spark e do Python disponíveis em um trabalho do AWS Glue para Spark. A versão do Python indica a versão compatível com trabalhos do tipo Spark. Não é assim que os ambientes de runtime do Ray são configurados.

Os trabalhos do Ray, você deve definir `GlueVersion` como `4.0` ou mais. Porém, as versões do Ray, do Python e das bibliotecas adicionais disponíveis no seu trabalho do Ray são determinadas pelo campo `Runtime` na definição do trabalho.

O ambiente de runtime to `Ray2.4` estará disponível por no mínimo 6 meses após o lançamento. À medida que o Ray evolui rapidamente, você poderá incorporar atualizações e melhorias do Ray por meio de versões futuras do runtime.

Valores válidos: `Ray2.4`

Valor do runtime	Versões do Ray e Python
`Ray2.4` (para AWS Glue 4.0+)	Ray 2.4.0 Python 3.9

Informações adicionais

- Para obter as notas de versão que acompanham os lançamentos do AWS Glue para Ray, consulte the section called “Versões do AWS Glue”.
- Para bibliotecas do Python que são fornecidas em um runtime, consulte the section called “Módulos fornecidos com trabalhos do Ray”.

Explicar os operadores em trabalhos do Ray

O AWS Glue executa trabalhos do Ray em novos tipos de operadores EC2 baseados em Graviton, que só estão disponíveis para trabalhos do Ray. Para provisionar adequadamente esses trabalhadores para workloads para os quais o Ray foi projetado, fornecemos uma proporção diferente de recursos de computação e recursos de memória da maioria dos operadores. Para processar esses recursos, usamos a unidade de processamento de dados otimizada para memória (M-DPU) em vez da unidade de processamento de dados padrão (DPU).

- Uma M-DPU corresponde a 4 vCPUs e 32 GB de memória.
- Uma DPU corresponde a 4 vCPUs e 16 GB de memória. As DPUs são usadas para contabilizar os recursos no AWS Glue com trabalhos do Spark e os operadores correspondentes.

Trabalhos do Ray têm acesso a um tipo de operador, `Z.2X`. O operador `Z.2X` mapeia para 2 M-DPUs (8 vCPUs, 64 GB de memória) e tem 128 GB de espaço em disco. Uma máquina `Z.2X` fornece 8 operadores Ray (um por vCPU).

O número de M-DPUs que você pode usar de forma concorrente em uma conta está sujeito a uma cota de serviço. Para obter mais informações sobre os limites da sua conta do AWS Glue, consulte endpoints e cotas do AWS Glue.

Você especifica o número de nós de processamento que estão disponíveis para um trabalho do Ray com `--number-of-workers (NumberOfWorkers)` na definição do trabalho. Para obter mais informações sobre os valores do Ray na API de trabalhos, consulte the section called “Tarefas”.

Você também pode especificar um número mínimo de operadores que um trabalho do Ray deve alocar com o parâmetro do trabalho `--min-workers`. Para obter mais informações sobre parâmetros de trabalho, consulte the section called “Referência”.

Usar parâmetros de trabalho em trabalhos do Ray

Você define os argumentos para os trabalhos do Ray no AWS Glue da mesma forma que define os argumentos no AWS Glue para trabalhos do Spark. Para obter mais informações sobre a API do AWS Glue, consulte the section called “Tarefas”. Você pode configurar trabalhos do Ray no AWS Glue com argumentos diferentes, que estão listados nesta referência. Você também pode fornecer seus próprios argumentos.

É possível configurar um trabalho pelo do console, na guia Job details (Detalhes do trabalho), no cabeçalho Job Parameters (Parâmetros do trabalho). Você também pode configurar um trabalho usando a AWS CLI definindo `DefaultArguments` em um trabalho ou definindo `Arguments` em uma execução de trabalho. Os argumentos padrão e os parâmetros do trabalho serão mantidos com o trabalho durante várias execuções.

Por exemplo, veja a seguir a sintaxe para a execução de um trabalho usando `--arguments` para definir um parâmetro especial.

```
$ aws glue start-job-run --job-name "CSV to CSV" --arguments='--scriptLocation="s3://
my_glue/libraries/test_lib.py",--test-environment="true"'
```

Depois de definir os argumentos, você pode acessar os parâmetros do trabalho de dentro do trabalho do Ray por meio das variáveis de ambiente. Essa é uma forma de configurar o trabalho para cada execução. O nome da variável de ambiente será o nome do argumento do trabalho sem o prefixo `--`.

No exemplo anterior, os nomes das variáveis seriam `scriptLocation` e `test-environment`. Depois, você recuperaria o argumento por meio dos métodos disponíveis na biblioteca padrão: `test_environment = os.environ.get('test-environment')`. Para obter mais informações sobre como acessar variáveis de ambiente com o Python, consulte módulo de sistema operacional na documentação do Python.

Configurar a forma como os trabalhos do Ray geram logs

Por padrão, os trabalhos do Ray geram logs e métricas que são enviados para o CloudWatch e para o Amazon S3. É possível usar o parâmetro `--logging_configuration` para alterar a forma como os logs são gerados. No momento, você pode usá-lo para impedir que os trabalhos do Ray gerem diversos tipos de logs. Este parâmetro usa um objeto em JSON, cujas chaves correspondem aos logs ou aos comportamentos que você gostaria de alterar. Ele oferece suporte para as seguintes chaves:

- `CLOUDWATCH_METRICS`: configura as séries de métricas do CloudWatch que podem ser usadas para visualizar a integridade do trabalho. Para obter mais informações sobre métricas, consulte the section called “Métricas de trabalho do Ray”.
- `CLOUDWATCH_LOGS`: configura os logs do CloudWatch que fornecem detalhes no nível da aplicação do Ray sobre o status de execução do trabalho. Para obter mais informações sobre logs, consulte the section called “Solucionar problemas de erros do Ray”.
- `S3`: configura o que o AWS Glue grava no Amazon S3, principalmente em relação a informações semelhantes aos logs do CloudWatch, mas como arquivos em vez de fluxos de logs.

Para desabilitar um comportamento de registro em log do Ray, forneça o valor `{\"IS_ENABLED\": \"False\"}`. Por exemplo, para desabilitar as métricas e os logs do CloudWatch, forneça a seguinte configuração:

```
"--logging_configuration": "{\"CLOUDWATCH_METRICS\": {\"IS_ENABLED\": \"False\"},
 \"CLOUDWATCH_LOGS\": {\"IS_ENABLED\": \"False\"}}"
```

Referência

Os trabalhos do Ray reconhecem os seguintes nomes de argumento que podem ser usados para configurar o ambiente de script para os trabalhos e execuções de trabalhos do Ray:

- `--logging_configuration`: é usado para interromper a geração de diversos logs criados por trabalhos do Ray. Esses logs são gerados por padrão em todos os trabalhos do Ray. Formato: objeto em JSON com escape de string. Para ter mais informações, consulte the section called “Configurar a forma como os trabalhos do Ray geram logs”.
- `--min-workers`: o número mínimo de nós de processamento alocados a um trabalho do Ray. Um nó de processamento pode executar várias réplicas, uma por cada CPU virtual. Formato: número inteiro. Mínimo: 0. Máximo: valor especificado em `--number-of-workers`

(`NumberOfWorkers`) na definição do trabalho. Para obter mais informações sobre a contabilização de nó de processamento, consulte the section called “Explicar os operadores em trabalhos do Ray”.

- `--object_spilling_config`: o AWS Glue para Ray permite o uso do Amazon S3 como forma de ampliar o espaço disponível para o armazenamento de objetos do Ray. Para habilitar esse comportamento, você pode fornecer ao Ray um objeto de configuração JSON Object spilling com esse parâmetro. Para obter mais informações sobre a configuração de object spilling do Ray, consulte Object Spilling na documentação do Ray. Formato: objeto JSON.

  O AWS Glue para Ray só permite spilling para o disco ou para o Amazon S3 de uma só vez. Você pode fornecer vários locais para spilling, desde que respeitem essa limitação. Ao fazer o spilling para o Amazon S3, você também precisará adicionar permissões do IAM ao trabalho para esse bucket.

  Ao fornecer um objeto JSON como configuração com a CLI, você deve fornecê-lo como uma string, com o objeto JSON escapado por string. Por exemplo, um valor de string para spilling para um caminho do Amazon S3 seria assim: `"{\"type\": \"smart_open\", \"params\": {\"uri\":\"s3path\"}}"`. No AWS Glue Studio, forneça esse parâmetro como um objeto JSON sem formatação extra.

- `--object_store_memory_head`: a memória alocada para o armazenamento de objetos Plasma no nó principal do Ray. Essa instância executa serviços de gerenciamento de cluster, bem como réplicas de processador. O valor representa uma porcentagem da memória livre na instância após uma inicialização a quente. Use esse parâmetro para ajustar as workloads com uso intenso de memória; os valores padrão são aceitáveis para a maioria dos usuários. Formato: número inteiro positivo. Mínimo: 1. Máximo: 100.

  Para obter mais informações sobre o Plasma, consulte The Plasma In-Memory Object Store (Armazenamento de objeto em memória do Plasma) na documentação do Ray.

- `--object_store_memory_worker`: a memória alocada para o armazenamento de objetos do Plasma no nó principal do Ray. Essas instâncias executam apenas réplicas de processador. O valor representa uma porcentagem da memória livre na instância após uma inicialização a quente. Esse parâmetro é usado para ajustar as workloads com uso intenso de memória; os valores padrão são aceitáveis para a maioria dos usuários. Formato: número inteiro positivo. Mínimo: 1. Máximo: 100.

  Para obter mais informações sobre o Plasma, consulte The Plasma In-Memory Object Store (Armazenamento de objeto em memória do Plasma) na documentação do Ray.

- `--pip-install`: um conjunto de pacotes do Python a serem instalados. Você pode instalar pacotes do PyPI usando esse argumento. Formato: lista delimitada por vírgulas.

  Uma entrada de pacote PyPI tem o formato de `package==version`, com o nome PyPI e a versão do pacote de destino. As entradas usam a correspondência de versões do Python para combinar o pacote com a versão, como ==, não um único sinal de igual =. Existem outros operadores que correspondem à versão. Para obter mais informações, consulte PEP 440 no site da Python. Você também pode fornecer os módulos personalizados com os `--s3-py-modules`.

- `--s3-py-modules`: um conjunto de caminhos do Amazon S3 que hospedam distribuições de módulos do Python. Formato: lista delimitada por vírgulas.

  Você pode usar essa opção para distribuir seus próprios módulos para o seu trabalho do Ray. Você também pode fornecer os módulos do PyPi com `--pip-install`. Ao contrário de ETL do AWS Glue, os módulos personalizados não são configurados por pip, mas são passados para o Ray para distribuição. Para ter mais informações, consulte the section called “Módulos adicionais do Python para trabalhos do Ray”.

- `--working-dir`: um caminho para um arquivo.zip hospedado no Amazon S3 que contém arquivos a serem distribuídos a todos os nós que executam o trabalho do Ray. Formato: string. Para ter mais informações, consulte the section called “Fornecer arquivos para o trabalho do Ray”.

Monitorar trabalhos do Ray com métricas

É possível monitorar trabalhos do Ray usando o AWS Glue Studio e o Amazon CloudWatch. O CloudWatch coleta e processa métricas brutas do AWS Glue com o Ray, o que as torna disponíveis para análise. Essas métricas são visualizadas no console do AWS Glue Studio, para que você possa monitorar o trabalho enquanto ele é executado.

Para obter uma visão geral de como monitorar o AWS Glue, consulte the section called “Usar métricas do Amazon CloudWatch”. Para obter uma visão geral de como usar as métricas do CloudWatch publicadas pelo AWS Glue, consulte the section called “Monitoramento com CloudWatch”.

Monitorar tarefas do Ray no console do AWS Glue

Na página de detalhes de uma execução de trabalho, abaixo da seção Detalhes da execução, você pode ver gráficos agregados pré-montados que mostram as métricas de trabalho disponíveis. O AWS Glue Studio envia as métricas de trabalho para o CloudWatch para toda execução de trabalho.

Com elas, você pode criar um perfil do cluster e das tarefas, bem como acessar informações detalhadas sobre cada nó.

Para obter mais informações sobre métricas gráficas disponíveis, consulte the section called "Visualizar métricas do Amazon CloudWatch para uma execução de trabalho do Ray".

Visão geral das métricas de trabalhos do Ray no CloudWatch

Publicamos métricas do Ray quando o monitoramento detalhado está ativado no CloudWatch. As métricas são publicadas no namespace `Glue/Ray` do CloudWatch.

- Métricas de instância

  Publicamos métricas sobre a utilização de CPU, memória e disco das instâncias designadas para uma tarefa. Essas métricas são identificadas por recursos como `ExecutorId`, `ExecutorType` e `host`. Essas métricas são um subconjunto das métricas padrão do agente Linux CloudWatch. Você pode encontrar informações sobre nomes e recursos de métricas na documentação do CloudWatch. Para obter mais informações, consulte Métricas coletadas pelo atendente do CloudWatch.

- Métricas de cluster do Ray

  Encaminhamos as métricas dos processos do Ray que executam o script nesse namespace, depois fornecemos a você as mais vitais. As métricas disponíveis podem diferir segundo a versão do Ray. Para obter mais informações sobre qual versão do Ray seu trabalho está executando, consulte the section called "Versões do AWS Glue".

  O Ray coleta métricas no nível da instância. Ele também fornece métricas para tarefas e para o cluster. Para obter mais informações sobre a estratégia de métricas subjacente do Ray, consulte Metrics na documentação do Ray.

> **Note**
>
> Não publicamos métricas de Ray no namespace do `Glue/Job Metrics/`, que é usado apenas para trabalhos do AWS Glue ETL.

Configurar propriedades de trabalho para trabalhos de shell Python no AWS Glue

É possível usar um trabalho de shell do Python para executar scripts do Python como um shell no AWS Glue. Com uma tarefa de shell do Python, é possível executar scripts compatíveis com Python 3.6 ou Python 3.9.

Tópicos

Limitações

Observe as seguintes limitações de trabalhos do shell do Python:

- Não é possível usar marcadores de trabalho com trabalhos de shell de Python.
- Não é possível empacotar bibliotecas do Python como arquivos `.egg` no Python 3.9+. Em seu lugar, use `.whl`.
- A opção `--extra-files` não pode ser usada devido a uma limitação de cópias temporárias de dados do S3.

Definir propriedades de trabalho para trabalhos de shell Python

Estas seções descrevem a definição das propriedades de trabalho no AWS Glue Studio ou usando a AWS CLI.

AWS Glue Studio

Ao definir o trabalho de shell do Python no AWS Glue Studio, você fornece algumas das seguintes propriedades:

IAM role (Perfil do IAM)

Especifique a função do AWS Identity and Access Management (IAM) usada para a autorização de recursos necessários para a execução do trabalho e acesso ao datastore. Para obter mais informações sobre permissões de execução de trabalho no AWS Glue, consulte Gerenciamento de identidade e acesso para AWS Glue.

Tipo

Selecione Python shell (Shell do Python) para executar um script do Python com o comando de trabalho chamado `pythonshell`.

Versão do Python

Escolha a versão do Python. O padrão é o Python 3.9. As versões válidas são Python 3.6 e Python 3.9.

Carregar bibliotecas comuns de análise (recomendado)

Escolha essa opção para incluir bibliotecas comuns para Python 3.9 no shell do Python.

Se suas bibliotecas forem personalizadas ou entrarem em conflito com as bibliotecas pré-instaladas, você poderá optar por não instalar bibliotecas comuns. No entanto, é possível instalar bibliotecas adicionais além das bibliotecas comuns.

Ao marcar essa opção, a opção `library-set` fica definida como `analytics`. Ao desmarcar essa opção, a opção `library-set` fica definida como `none`.

Nome do arquivo de script e caminho do script

O código no script define a lógica processual do trabalho. É possível fornecer o nome e o local do script no Amazon Simple Storage Service (Amazon S3). Confirme se não existe um arquivo com o mesmo nome que o diretório do script no caminho. Para saber mais sobre como usar scripts, consulte Guia de programação do AWS Glue.

Script

O código no script define a lógica processual do trabalho. Você pode codificar o script em Python 3.6 ou Python 3.9. Você pode editar um script no AWS Glue Studio.

Unidades de processamento de dados

O número máximo de unidades de processamento de dados (DPUs) do AWS Glue que poderão ser alocadas quando esse trabalho for executado. Uma DPU é uma medida relativa do poder

de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte Preços do AWS Glue.

Você pode definir o valor como 0,0625 ou 1. O padrão é 0.0625. Em ambos os casos, o disco local da instância será de 20 GB.

CLI

Também é possível criar um trabalho de Python shell (Shell do Python) usando a AWS CLI, como no exemplo a seguir.

```
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**
>
> Você não precisa especificar a versão do AWS Glue, pois o parâmetro `--glue-version` não se aplica a trabalhos de shell do AWS Glue. Qualquer versão especificada será ignorada.

Por padrão, os trabalhos que você cria com a AWS CLI são definidos para Python 3. As versões válidas do Python são 3 (correspondendo à versão 3.6) e 3.9. Para especificar o Python 3.6, adicione a seguinte tupla ao parâmetro `--command`: `"PythonVersion":"3"`

Para especificar o Python 3.9, adicione a seguinte tupla ao parâmetro `--command`: `"PythonVersion":"3.9"`

Para definir a capacidade máxima usada por um trabalho de shell do Python, forneça o parâmetro `--max-capacity`. Para trabalhos de shell do Python, o parâmetro `--allocated-capacity` não pode ser usado.

Bibliotecas compatíveis com trabalhos de shell Python

Ao usar o Python 3.9, você pode escolher no shell do Python o conjunto de bibliotecas para usar conjuntos pré-empacotados de bibliotecas conforme suas necessidades. Você pode usar a opção `library-set` para escolher o conjunto de bibliotecas. Os valores válidos são `analytics` e `none`.

O ambiente para executar um trabalho de shell de Python é compatível com as seguintes bibliotecas:

Versão do Python	Python 3.6	Python 3.9	
Conjunto de bibliotecas	N/D	analytics	none
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	

Versão do Python	Python 3.6	Python 3.9	
redshift-connector		2.0.907	
solicitações	2.22.0	2.27.1	
scikit-learn	0.20.3	1.0.2	
scipy	1.2.1	1.8.0	
SQLAlchemy		1.4.36	
s3fs		2022.3.0	

Você pode usar a biblioteca `NumPy` em um trabalho de shell do Python para computação científica. Para obter mais informações, consulte NumPy. O exemplo a seguir mostra um script NumPy que pode ser usado em um trabalho de shell do Python. O script imprime "Hello world" e os resultados de vários 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)
```

Fornecer sua própria biblioteca Python

Usar o PIP

Ao usar o Python 3.9, o shell do Python permite que você forneça módulos adicionais ou versões diferentes do Python por trabalho. Você pode usar a opção `--additional-python-modules` com uma lista de módulos do Python separados por vírgulas para adicionar um novo módulo ou alterar a versão de um existente. Você não pode fornecer módulos Python personalizados hospedados no Amazon S3 com esse parâmetro ao usar trabalhos de shell do Python.

Por exemplo, para atualizar ou adicionar um novo módulo `scikit-learn`, use o seguinte par de chave e valor: `"--additional-python-modules", "scikit-learn==0.21.3"`.

O AWS Glue usa o Python Package Installer (pip3) para instalar os módulos adicionais. Você pode passar opções adicionais de pip3 dentro do valor `--additional-python-modules`. Por exemplo, `"scikit-learn==0.21.3 -i https://pypi.python.org/simple/"`. Quaisquer incompatibilidades ou limitações do pip3 se aplicam.

> **Note**
>
> Para evitar incompatibilidades futuras, recomendamos o uso de bibliotecas compiladas para o Python 3.9.

Usar um arquivo Egg ou Whl

Talvez você já tenha uma ou mais bibliotecas Python empacotadas como um arquivo `.egg` ou `.whl`. Se esse for o caso, você poderá especificá-las para o trabalho usando a AWS Command Line Interface (AWS CLI) no sinalizador `"--extra-py-files"`, como no exemplo a seguir.

```
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"]}'
```

Se você não tiver certeza de como criar um arquivo `.egg` ou `.whl` de uma biblioteca Python, use as etapas a seguir. Este exemplo é aplicável a macOS, Linux e Windows Subsystem for Linux (WSL).

Como criar um arquivo .egg ou .whl em Python

1. Crie um cluster do Amazon Redshift em uma nuvem privada virtual (VPC) e adicione alguns dados a uma tabela.
2. Crie uma conexão do AWS Glue para a combinação VPC-GrupodeSegurança-Sub-rede que você usou para criar o cluster. Teste se a conexão foi bem-sucedida.
3. Crie um diretório chamado `redshift_example` e crie um arquivo chamado `setup.py`. Cole o seguinte código em `setup.py`.

```
from setuptools import setup

setup(
    name="redshift_module",
    version="0.1",
    packages=['redshift_module']
)
```

4. No diretório `redshift_example`, crie um diretório `redshift_module`. No diretório `redshift_module`, crie os arquivos `__init__.py` e `pygresql_redshift_common.py`.
5. Deixe o campo `__init__.py` vazio. Em `pygresql_redshift_common.py`, cole o seguinte código: Substitua *`port`* (porta), *`db_name`* (nome do bd), *`user`* (usuário) e *`password_for_user`* (senha do usuário) por detalhes específicos do seu cluster do Amazon Redshift. Substitua *`table_name`* (nome da tabela) pelo nome da tabela no 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. Se você ainda não estiver lá, altere para o diretório `redshift_example`.
7. Execute um destes procedimentos:

   - Execute o seguinte comando para criar um arquivo `.egg`.

     ```
     python setup.py bdist_egg
     ```

   - Execute o seguinte comando para criar um arquivo `.whl`.

     ```
     python setup.py bdist_wheel
     ```

8. Instale as dependências necessárias para o comando anterior.
9. O comando cria um arquivo no diretório `dist`:

   - Se você criou um arquivo egg, ele se chama `redshift_module-0.1-py2.7.egg`.
   - Se você criou um arquivo wheel, ele se chama `redshift_module-0.1-py2.7-none-any.whl`.

   Faça upload desse arquivo no Amazon S3.

   Neste exemplo, o caminho do arquivo carregado é *`s3://DOC-EXAMPLE-BUCKET/EGG-FILE`* ou *`s3://DOC-EXAMPLE-BUCKET/WHEEL-FILE`*.

10. Crie um arquivo Python para ser usado como um script para o trabalho do AWS Glue e adicione o seguinte código ao arquivo.

    ```
    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. Faça o upload do arquivo anterior no Amazon S3. Neste exemplo, o caminho do arquivo carregado é *`s3://DOC-EXAMPLE-BUCKET/scriptname.py`*.

12. Crie um trabalho de shell do Python usando esse script. No console do AWS Glue, na página Job properties (Propriedades do trabalho), especifique o caminho para o arquivo `.egg/.whl` na caixa Python library path (Caminho da biblioteca Python). Se você tiver vários arquivos `.egg/.whl` e arquivos Python, forneça uma lista separada por vírgulas nesta caixa.

    Ao modificar ou renomear arquivos `.egg`, os nomes de arquivo devem usar os nomes padrão gerados pelo comando "python setup.py bdist_egg" ou devem aderir às convenções de nomenclatura do módulo Python. Para obter mais informações, consulte o Style Guide for Python Code.

    Usando a AWS CLI, crie um trabalho com um comando, como no exemplo a seguir.

    ```
    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"]}'
    ```

    Quando o trabalho é executado, o script imprime as linhas criadas na tabela *`table_name`* no cluster do Amazon Redshift.

Use AWS CloudFormation com trabalhos de shell Python no AWS Glue

Você pode usar AWS CloudFormation com trabalhos de shell Python no AWS Glue. Veja um exemplo a seguir:

```
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
```

O grupo do Amazon CloudWatch Logs para a saída de trabalhos de shell do Python é `/aws-glue/python-jobs/output`. Para erros, consulte o grupo de logs `/aws-glue/python-jobs/error`.

Como monitorar o AWS Glue

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e a performance do AWS Glue e das suas outras soluções da AWS. O AWS fornece ferramentas de monitoramento que você pode usar para observar o AWS Glue, informar quando algo está errado e realizar uma ação automaticamente quando apropriado:

Você pode usar as seguintes ferramentas de monitoramento automatizadas para observar o AWS Glue e gerar relatórios quando algo estiver errado:

- O Amazon CloudWatch Events oferece uma transmissão quase em tempo real de eventos do sistema que descrevem as mudanças nos recursos da AWS. O CloudWatch Events permite a computação automatizada baseada em eventos. Você pode escrever regras que observam certos eventos e acionam ações automatizadas em outros serviços da AWS quando esses eventos ocorrem. Para obter mais informações, consulte o Manual do usuário do Amazon CloudWatch Events.
- O Amazon CloudWatch Logs permite monitorar, armazenar e acessar os arquivos de log de instâncias do Amazon EC2, do AWS CloudTrail e de outras fontes. O CloudWatch Logs pode monitorar informações nos arquivos de log e notificar você quando determinados limites forem atingidos. É possível também arquivar seus dados de log em armazenamento resiliente. Para obter mais informações, consulte o Guia do usuário do Amazon CloudWatch Logs.
- O AWS CloudTrail captura chamadas de API e eventos relacionados feitos por, ou em nome de, sua conta da AWS e entrega os arquivos de log a um bucket do Amazon S3 que você especificar. Você pode identificar quais usuários e contas chamaram a AWS, o endereço IP de origem do qual as chamadas foram feitas e quando elas ocorreram. Para mais informações, consulte o Guia do usuário do AWS CloudTrail.

Além disso, você tem acesso aos seguintes insights no console do AWS Glue para ajudar você a depurar e criar perfis de trabalho:

- Trabalhos do Spark: é possível visualizar séries de métricas selecionadas do CloudWatch, e trabalhos mais recentes têm acesso à interface do usuário do Spark. Para ter mais informações, consulte the section called “Monitorar trabalhos Spark”.

- Trabalhos do Ray: é possível visualizar séries de métricas selecionadas do CloudWatch. Para ter mais informações, consulte the section called “Métricas de trabalho do Ray”.

Tópicos

Etiquetas da AWS no AWS Glue

Para ajudar você a gerenciar seus recursos do AWS Glue, você pode, opcionalmente, atribuir suas próprias tags para alguns tipos de recursos do AWS Glue. Uma tag é um rótulo atribuído a um recurso da AWS. Cada tag consiste em uma chave e um valor opcional, ambos definidos por você. Use tags no AWS Glue para organizar e identificar seus recursos. As tags podem ser usadas para criar relatórios de contabilidade de custos e restringir o acesso aos recursos. Se você estiver usando o AWS Identity and Access Management, pode controlar quais usuários na sua conta da AWS têm permissão para criar, editar ou excluir tags. Além das permissões para chamar as APIs relacionadas a tags, você também precisa da permissão `glue:GetConnection` para chamar APIs de marcação em conexões e da permissão `glue:GetDatabase` para chamar APIs de marcação em bancos de dados. Para ter mais informações, consulte ABAC com AWS Glue.

No AWS Glue, você pode marcar os seguintes recursos:

- Conexão
- Banco de dados
- Crawler
- Sessão interativa
- Endpoint de desenvolvimento
- Trabalho
- Trigger
- Fluxo de trabalho
- Blueprint
- Transformação de machine learning

- Regras de qualidade de dados
- Esquemas de fluxo
- Registros do esquema de fluxo

> **Note**
>
> Como uma das melhores práticas, para permitir a marcação desses recursos do AWS Glue, sempre inclua a ação `glue:TagResource` em suas políticas.

Considere o seguinte ao usar tags no AWS Glue.

- Uma entidade suporta um máximo de 50 tags.
- No AWS Glue, você especifica as tags como uma lista de pares de chave-valor no formato `{"string": "string" ...}`
- Quando você cria uma tag em um objeto, a chave da tag é obrigatória, e o valor da tag é opcional.
- A chave e o valor da tag diferenciam maiúsculas de minúsculas.
- A chave da tag e o valor da tag não devem conter o prefixo aws. Nenhuma operação é permitida nessas tags.
- O comprimento máximo da chave da tag é de 128 caracteres Unicode em UTF-8. A chave da tag não pode estar vazia nem ser nula.
- O comprimento máximo do valor da tag é de 256 caracteres Unicode em UTF-8. O valor da tag pode estar vazio ou ser nulo.

Suporte para marcação para conexões do AWS Glue

Você pode restringir as permissões das ações `CreateConnection`, `UpdateConnection`, `GetConnection` e `DeleteConnection` com base na marcação de recurso. Isso permite que você implemente o controle de acesso de privilégio mínimo em trabalhos do AWS Glue com fontes de dados JDBC que precisam buscar informações de conexão JDBC no catálogo de dados.

Exemplo de uso

Crie uma conexão do AWS Glue com a marcação ["connection-category", "dev-test"].

Especifique a condição da marcação para a ação `GetConnection` na política do IAM.

```
{
   "Effect": "Allow",
   "Action": [
     "glue:GetConnection"
     ],
   "Resource": "*",
   "Condition": {
     "ForAnyValue:StringEquals": {
       "aws:ResourceTag/tagKey": "dev-test"
     }
   }
 }
```

Exemplos

Os exemplos a seguir criam um trabalho com tags atribuídas.

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
```

JSON do AWS CloudFormation

```
{
  "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"
           }
```

```
        }
      }
    }
  }
}
```

YAML do AWS CloudFormation

```
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 obter mais informações, consulte Estratégias de marcação da AWS.

Para obter informações sobre como controlar o acesso usando tags, consulte ABAC com AWS Glue.

Automatizando o AWS Glue com o CloudWatch Events

Você pode usar o Amazon CloudWatch Events para automatizar seus produtos da AWS e responder automaticamente a eventos do sistema, como problemas de disponibilidade de aplicação ou alterações de recursos. Os eventos dos produtos da AWS são entregues ao CloudWatch Events quase em tempo real. Você pode escrever regras simples para indicar quais eventos são do seu interesse, e as ações automatizadas a serem tomadas quando um evento corresponder à regra. As ações que podem ser automaticamente acionadas incluem as seguintes:

- Invocar uma função do AWS Lambda
- Invocar o comando de execução do Amazon EC2
- Transmitir o evento Amazon Kinesis Data Streams
- Ativação de uma máquina de estado do AWS Step Functions
- Notificar um tópico do Amazon SNS ou uma fila do Amazon SQS

Alguns exemplos de uso do CloudWatch Events com o AWS Glue incluem os seguintes:

- Ativar uma função Lambda quando um trabalho de ETL é executado com sucesso
- Notificar um tópico do Amazon SNS quando um trabalho de ETL apresentar falha

Os seguintes CloudWatch Events são gerados pelo AWS Glue.

- Os eventos de `"detail-type":"Glue Job State Change"` são gerados para `SUCCEEDED`, `FAILED`, `TIMEOUT` e `STOPPED`.
- Os eventos de `"detail-type":"Glue Job Run Status"` são gerados para execuções de trabalho de `RUNNING` e de `STARTINGSTOPPING`, quando elas excedem o limite da notificação de

atraso do trabalho. Você deve definir a propriedade de limite de notificação de atraso de trabalho para receber esses eventos.

Quando o limite de notificação de atraso de trabalho é excedido, apenas um único evento é gerado por status de execução de trabalho.

- Os eventos de `"detail-type":"Glue Crawler State Change"` são gerados por `Started`, `Succeeded` e `Failed`.
- Os eventos de `"detail-type":"Glue Data Catalog Database State Change"` são gerados para `CreateDatabase`, `DeleteDatabase`, `CreateTable`, `DeleteTable` e `BatchDeleteTable`. Por exemplo, se uma tabela for criada ou excluída, será enviada uma notificação ao CloudWatch Events. Observe que não é possível escrever um programa que depende da ordem ou da existência de eventos de notificação, já que eles podem estar fora de sequência ou ausentes. Os eventos são emitidos com base no melhor esforço. Nos detalhes da notificação:
  - O `typeOfChange` contém o nome da operação da API.
  - O `databaseName` contém o nome do banco de dados afetado.
  - O `changedTables` contém até 100 nomes de tabelas afetadas por notificação. Quando os nomes das tabelas forem muito longos, poderão ser criadas várias notificações.
- Os eventos de `"detail-type":"Glue Data Catalog Table State Change"` são gerados para `UpdateTable`, `CreatePartition`, `BatchCreatePartition`, `UpdatePartition`, `DeletePartition`, `BatchUpdatePartition` e `BatchDeletePartition`. Por exemplo, se uma tabela ou partição for atualizada, será enviada uma notificação ao CloudWatch Events. Observe que não é possível escrever um programa que depende da ordem ou da existência de eventos de notificação, já que eles podem estar fora de sequência ou ausentes. Os eventos são emitidos com base no melhor esforço. Nos detalhes da notificação:
  - O `typeOfChange` contém o nome da operação da API.
  - O `databaseName` contém o nome do banco de dados que contém os recursos afetados.
  - O `tableName` contém o nome da tabela afetada.
  - O `changedPartitions` especifica até 100 partições afetadas em uma notificação. Quando os nomes das partições forem muito longos, poderão ser criadas várias notificações.

    Por exemplo, se houver duas chaves de partição, `Year` e `Month`, `"2018,01", "2018,02"` modificará a partição em que `"Year=2018" and "Month=01"` e a partição em que `"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 obter mais informações, consulte o Manual do usuário do Amazon CloudWatch Events. Para eventos específicos do AWS Glue, consulte Eventos do AWS Glue.

Monitoramento de recursos do AWS Glue

O AWS Glue tem limites de serviço para proteger os clientes de provisionamento excessivo inesperado e de ações maliciosas destinadas a aumentar sua fatura. Os limites também protegem o serviço. Ao fazer login no console do AWS Service Quota, os clientes podem visualizar seus limites atuais de recursos e solicitar um aumento (quando apropriado).

O AWS Glue permite que você visualize o uso de recursos do serviço como uma porcentagem no Amazon CloudWatch e configure alarmes do CloudWatch para monitorar o uso. O Amazon CloudWatch fornece monitoramento de recursos da AWS e de aplicações dos clientes sem execução na infraestrutura da Amazon. As métricas são gratuitas para você. As seguintes métricas têm suporte:

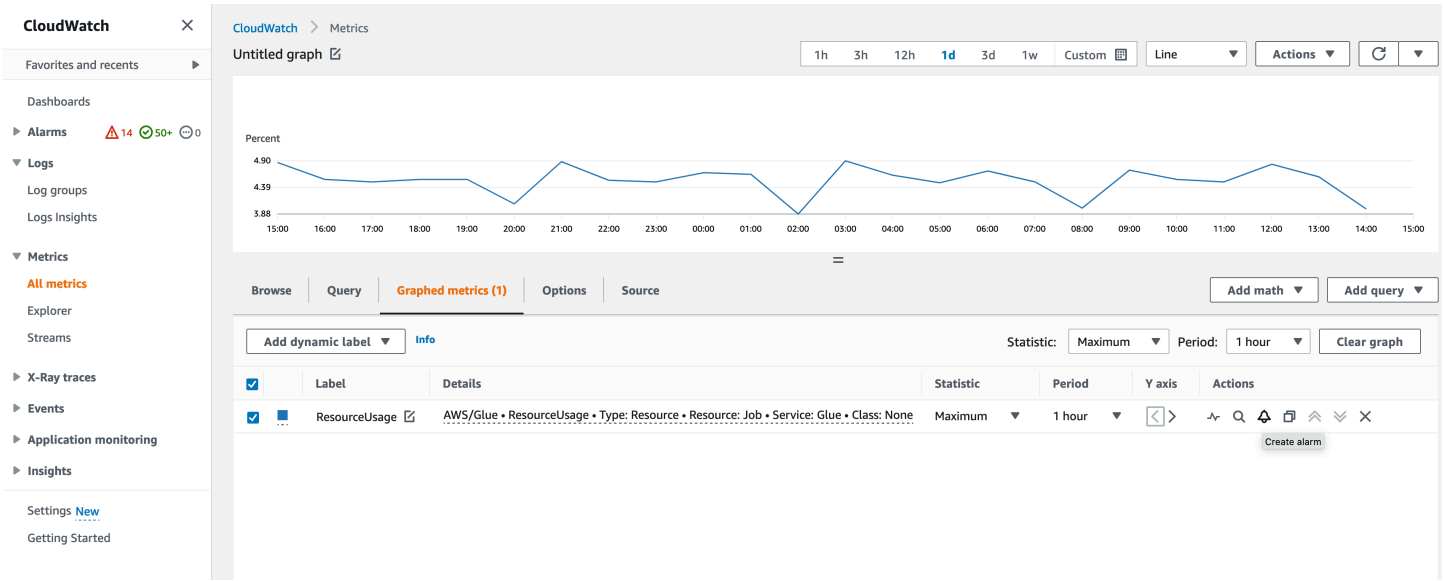
- Número de fluxos de trabalho por conta
- Número de gatilhos por conta
- Número de jobs por conta

- Número de execuções de trabalho simultâneas por conta
- Número de esquemas por conta
- Número de sessões interativas por conta

Configurar e usar métricas de recursos

Para usar esse atributo, você pode acessar o console do Amazon CloudWatch para visualizar as métricas e configurar os alarmes. As métricas estão no namespace AWS/Glue e são uma porcentagem da contagem real de uso do recurso dividida pela cota do recurso. As métricas do CloudWatch são entregues às suas contas, e não terão nenhum custo para você. Por exemplo, se você tiver 10 fluxos de trabalho criados e sua cota de serviço permitir que você tenha 200 fluxos de trabalho no máximo, seu uso será 10/200 = 5% e, no gráfico, você verá um ponto de dados de 5 como porcentagem. Para ser mais específico:

```
Namespace: AWS/Glue
Metric name: ResourceUsage
Type: Resource
Resource: Workflow (or Trigger, Job, JobRun, Blueprint, InteractiveSession)
Service: Glue
Class: None
```



Para criar um alarme com base em uma métrica no console do CloudWatch:

1. Depois de localizar a métrica, acesse Métricas em gráfico.

2. Clique em Criar alarme em Ações.
3. Configure o alarme conforme necessário.

Nós emitimos métricas sempre que seu uso de recursos sofrer uma alteração (como um aumento ou diminuição). Mas se o uso de seus recursos não mudar, emitimos métricas de hora em hora, para que você tenha um gráfico contínuo do CloudWatch. Para evitar a perda de pontos de dados, não recomendamos que você configure um período inferior a 1 hora.

Você também pode configurar alarmes usando o AWS CloudFormation como no exemplo a seguir. Neste exemplo, quando o uso de recursos do fluxo de trabalho atinge 80%, ele aciona um alarme para enviar uma mensagem para o tópico existente do SNS, onde você pode se inscrever para receber notificações.

```
{
 "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"
  }
}
```

Registrar em log chamadas de API do AWS Glue com o AWS CloudTrail

O AWS Glue é integrado a AWS CloudTrail, serviço que fornece um registro das ações realizadas por um usuário, perfil ou AWS serviço em AWS Glue. O CloudTrail captura todas as chamadas API para AWS Glue como eventos. As chamadas capturadas incluem as aquelas do AWS Glue console e chamadas de código para AWS Glue operações API. Caso crie uma trilha, voce pode habilitar a entrega contínua de eventos CloudTrail para um bucket Amazon S3, inclusive eventos para AWS Glue. Se você não configurar uma trilha, ainda poderá visualizar os eventos mais recentes no console do CloudTrail em Histórico de eventos. Ao fazer uso das informações coletadas pelo CloudTrail, é possível determinar a solicitação feita a AWS Glue, o endereço IP no qual a solicitação foi feita, quem fez a solicitação e quando foi feita, além de detalhes adicionais.

Para saber mais sobre o CloudTrail, consulte o AWS CloudTrail Guia de Usuário.

Informações do AWS Glue no CloudTrail

O CloudTrail é ativado na sua conta da AWS quando ela é criada. Quando ocorre uma atividade no AWS Glue, essa atividade é registrada em um evento do CloudTrail com outros eventos de produtos da AWS em Histórico de eventos. É possível visualizar, pesquisar e baixar os eventos recentes na sua conta da AWS. Para mais informações, consulte Visualizando Eventos com Histórico de Eventos CloudTrail.

Para um registro contínuo de eventos em sua AWS conta, inclusive eventos para AWS Glue, crie uma trilha. Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as Regiões da AWS. A trilha registra em log eventos de todas as Regiões na partição da AWS e entrega os arquivos de log para o bucket do Amazon S3 especificado por você. Além disso, é possível configurar outros AWS serviços para melhor analisar e agir de acordo com dados coletados do evento nos logs do CloudTrail. Para mais informações, consulte:

- Criar uma trilha para sua conta da AWS
- Serviços e integrações compatíveis com o CloudTrail
- Configurando Notificações Amazon SNS para CloudTrail
- Receber arquivos de log do CloudTrail de várias regiões e receber arquivos de log do CloudTrail de várias contas

Todas as ações do AWS Glue são registradas pelo CloudTrail e são documentadas em AWS Glue API. Por exemplo, as chamadas às ações `CreateDatabase`, `CreateTable` e `CreateScript` geram entradas nos arquivos de log do CloudTrail.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário raiz ou do usuário do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro serviço da AWS.

Para obter mais informações, consulte o Elemento userIdentity do CloudTrail.

No entanto, o CloudTrail não registra em log todas as informações relacionadas às chamadas. Por exemplo, ele não registra em log certas informações confidenciais, como `ConnectionProperties` usadas com as solicitações e, em vez disso, registra o valor `null` nas respostas retornadas pelas seguintes APIs:

```
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
```

Noções básicas sobre entradas de arquivos de log do AWS Glue

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log a um bucket do Amazon S3 especificado. Os arquivos de log CloudTrail contêm uma ou mais entradas de log.

Um evento representa uma única solicitação de qualquer fonte, e inclui informações sobre a ação solicitada, data e hora da ação, parâmetros de solicitação e assim por diante. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API pública, portanto não são exibidos em uma ordem específica.

O exemplo a seguir mostra uma entrada de log CloudTrail que demonstra a `DeleteCrawler` ação.

```
{
  "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"
}
```

O exemplo mostra uma entrada de log do CloudTrail que demonstra a ação `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"
}
```

Status de execução de trabalhos do AWS Glue

Você pode visualizar o status de um trabalho de extração, transformação e carregamento (ETL) do AWS Glue enquanto ele está em execução ou após a interrupção. Você pode visualizar o status usando o console do AWS Glue, a AWS Command Line Interface (AWS CLI) ou a ação GetJobRun na API do AWS Glue.

Os possíveis status de execução de trabalho são `STARTING`, `RUNNING`, `STOPPING`, `STOPPED`, `SUCCEEDED`, `FAILED`, `ERROR`, `WAITING` e `TIMEOUT`.

A tabela a seguir relaciona os status que indicam o término incomum de um trabalho.

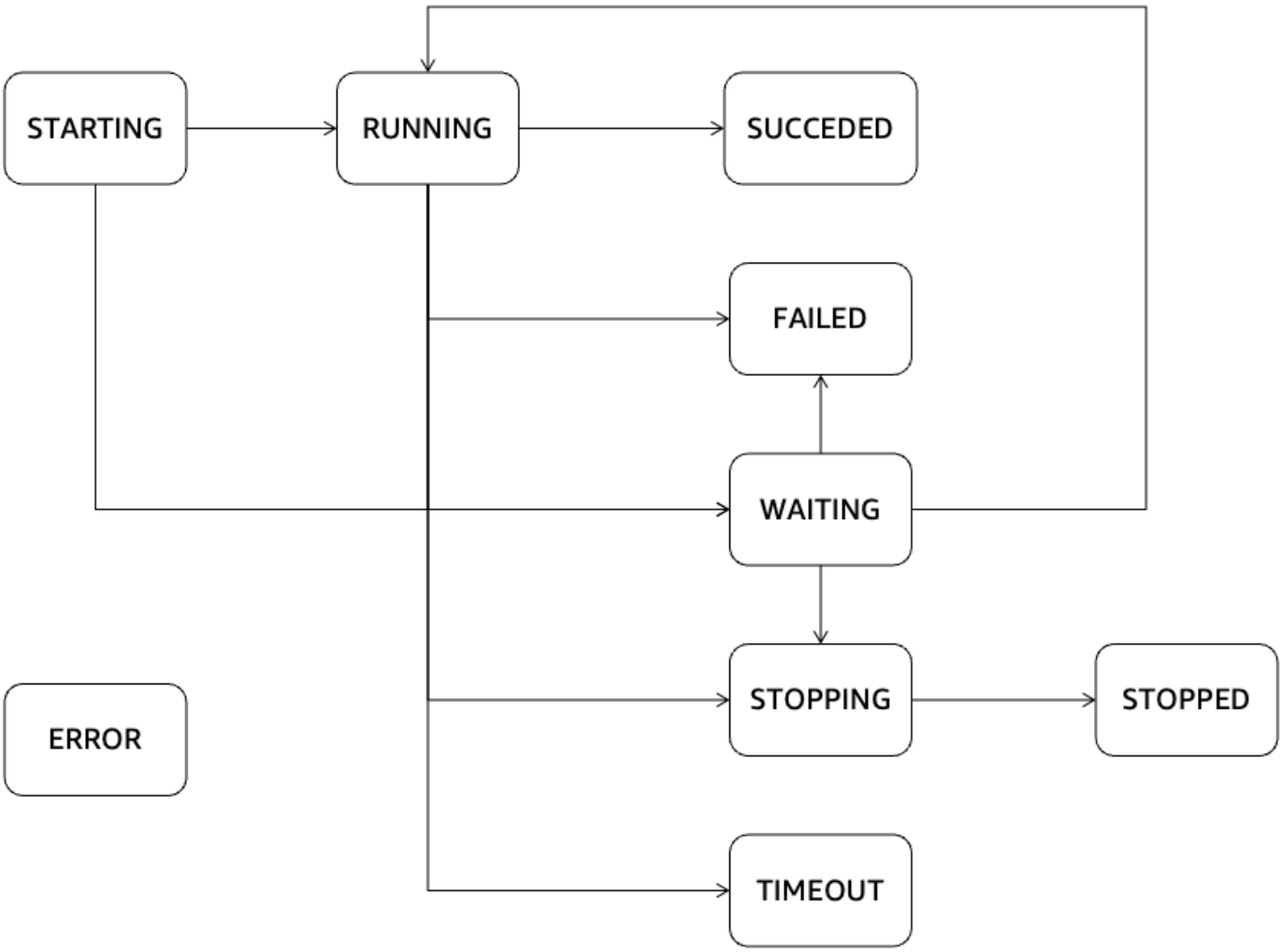
Status de execução de trabalho	Descrição
`FAILED`	O trabalho excedeu suas execuções simultâne as máximas permitidas ou terminou com um código de saída desconhecido.
`ERROR`	Um fluxo de trabalho, um acionador de programação ou de evento tentou executar um trabalho excluído.
`TIMEOUT`	O tempo de execução do trabalho excedeu seu valor de tempo limite especificado.

O status `WAITING` indica que a execução de um trabalho está aguardando recursos. A tabela a seguir descreve o comportamento de espera para diferentes classes de trabalhos.

Tipo de trabalho	Comportamento
Trabalhos do Spark (padrão)	Os trabalhos que não foram configurados para serem repetidos com base na sua configura ção de `maxRetries` podem entrar no estado AGUARDANDO. Uma nova execução de trabalho permanecerá no estado AGUARDAND O se o serviço não conseguir adquirir recursos suficientes para iniciar a execução. Isso poderá ocorrer devido às cotas de serviço da sua conta ou aos limites de capacidade em sua região encontrarem um dos seguintes casos de erro: • O máximo de execuções simultâneas de trabalhos por conta foi excedido • Máximo de execuções simultâneas de trabalhos por trabalho foi excedido (inclui a cota de serviço no nível da conta, bem como o limite especificado no trabalho com `MaxConcurrentRuns`)

Tipo de trabalho	Comportamento
	• Computação simultânea máxima (uso de DPU) excedida • Recurso indisponível Para obter mais informações sobre as cotas de do serviço AWS Glue, consulte Endpoints e cotas do AWS Glue. O tempo que o AWS Glue aguardará por recursos poderá ser diferente dependendo das circunstâncias. Um trabalho pode fazer a transição entre status não terminal à medida que tenta adquirir recursos. Eventualmente, o trabalho será colocado no status `FAILED` se ele não conseguir adquirir recursos. AWS O Glue tentará novamente por no máximo 15 minutos ou 10 tentativas, o que ocorrer primeiro.
Trabalhos do Spark (Flex)	Uma nova execução de trabalho ficará no estado WAITING (AGUARDANDO) se o serviço não conseguir adquirir recursos suficientes para iniciar a execução, o que atrasará o início da execução. A execução ficará no estado WAITING (AGUARDANDO) por até 20 minutos (tempo limite controlad o pelo serviço). Após 15 minutos, o serviço tentará fazer uma inicialização forçada e, dependendo da capacidade disponível, a execução poderá iniciar ou falhar com uma mensagem de erro adequada.
Trabalhos de shell Python	Mesmo comportamento dos trabalhos padrão usando o Spark.

O diagrama de estado a seguir descreve as transições de estado esperadas ao longo do ciclo de vida de um trabalho do AWS Glue. Essas informações são aplicáveis a todos os tipos de trabalho.



AWS Glue Streaming

AWS Glue O streaming, um componente do AWS Glue, permite que você gerencie com eficiência os dados de streaming quase em tempo real, capacitando você a realizar tarefas cruciais, como ingestão, processamento e aprendizado de máquina de dados. Usando a estrutura Apache Spark Streaming, o AWS Glue Streaming fornece um serviço sem servidor que pode lidar com dados de streaming em grande escala. AWS Glue fornece várias otimizações além do Apache Spark, como infraestrutura sem servidor, auto-scaling, desenvolvimento visual de tarefas, notebooks instantâneos para trabalhos de streaming e outras melhorias de desempenho.

Casos de uso para o streaming

Alguns casos de uso comuns do AWS Glue streaming incluem:

N processamento de ear-real-time dados: o AWS Glue streaming permite que as organizações processem dados de streaming quase em tempo real, permitindo que obtenham insights e tomem decisões oportunas com base nas informações mais recentes.

Detecção de fraudes: você pode utilizar o AWS Glue Streaming para análise em tempo real dos dados de streaming, tornando-os valiosos para detectar atividades fraudulentas, como fraudes com cartões de crédito, intrusão na rede ou fraudes on-line. Ao processar e analisar continuamente os dados de entrada, você pode identificar rapidamente padrões ou anomalias suspeitas.

Análise de mídia social: o AWS Glue streaming pode processar dados de mídia social em tempo real, como tweets, postagens ou comentários, permitindo que as organizações monitorem tendências, analisem sentimentos e gerenciem a reputação da marca em tempo real.

Análise da Internet das Coisas (IoT): o AWS Glue streaming é adequado para lidar e analisar fluxos de dados de alta velocidade gerados por dispositivos de IoT, sensores e máquinas conectadas. Ele permite o monitoramento em tempo real, a detecção de anomalias, a manutenção preditiva e outros casos de uso de análise da IoT.

Análise de fluxo de cliques: o AWS Glue streaming pode processar e analisar dados de fluxo de cliques em tempo real de sites ou aplicativos móveis. Isso possibilita que as empresas obtenham insights sobre o comportamento do usuário, personalizem as experiências do usuário e otimizem campanhas de marketing com base em dados de fluxo de cliques em tempo real.

Monitoramento e análise de registros: o AWS Glue streaming pode processar e analisar continuamente dados de log de servidores, aplicativos ou dispositivos de rede em tempo real. Isso

ajuda a detectar anomalias, solucionar problemas e monitorar a integridade e a performance do sistema.

Sistemas de recomendação: o AWS Glue streaming pode processar dados de atividades do usuário em tempo real e atualizar os modelos de recomendação dinamicamente. Isso permite recomendações personalizadas e em tempo real com base no comportamento e nas preferências do usuário.

Esses são alguns exemplos dos diversos casos de uso em que o AWS Glue Streaming pode ser aplicado. Sua integração com o AWS ecossistema e os serviços gerenciados o torna uma opção conveniente para processamento e análise de streams em tempo real na nuvem.

Quais são os benefícios de usar o AWS Glue Streaming?

Os benefícios de usar o AWS Glue Streaming são os seguintes:

- Sem servidor: o AWS Glue streaming é sem servidor, eliminando a necessidade de gerenciar a infraestrutura. Isso reduz a sobrecarga operacional e permite que os usuários se concentrem nas tarefas de processamento e de análise de dados, em vez de no gerenciamento da infraestrutura.
- Escalonamento automático: o AWS Glue streaming fornece recursos de escalonamento automático, ajustando dinamicamente a capacidade de processamento com base na carga de trabalho. Ele aumenta ou reduz a escala horizontalmente de forma automática para lidar com as flutuações no volume de dados, garantindo uma performance e uma utilização de recursos ideais.
- Desenvolvimento visual: o streaming do desenvolvimento de tarefas pode ser complexo. AWS Glue O streaming aborda esse desafio oferecendo o AWS Glue Studio, uma ferramenta de criação visual. AWS Glue O Studio simplifica o processo de criação de fluxos de trabalho de streaming e permite que os desenvolvedores projetem e gerenciem aplicativos de streaming visualmente, reduzindo a curva de aprendizado e aumentando a produtividade.
- Econômico: como um serviço sem servidor, o AWS Glue streaming oferece eficiência de custos ao eliminar a necessidade de provisionamento e manutenção da infraestrutura. Os usuários são cobrados com base nos recursos consumidos durante a execução de trabalhos de streaming, permitindo a otimização de custos e a escalabilidade com base no uso real.
- Lida com cargas de trabalho complexas: o AWS Glue streaming foi projetado para lidar com cargas de trabalho de streaming complexas. Ele pode processar e analisar grandes volumes de dados em tempo real, suportar transformações avançadas e integrar-se a outros AWS serviços, permitindo fluxos de trabalho de análise e pipelines de dados de streaming sofisticados.

- Sem dependência: o AWS Glue streaming oferece flexibilidade e evita a dependência do fornecedor. Os usuários podem aproveitar o AWS Glue streaming como parte de um AWS ecossistema mais amplo, integrando-o perfeitamente a outros AWS serviços. Isso permite a fácil integração com fontes de dados, aplicações e serviços existentes, sem a necessidade de estar vinculado a uma tecnologia ou plataforma específica.

Quando usar o AWS Glue Streaming?

Existem muitas opções quando se trata de casos de uso de streaming. Recomendamos AWS Glue fazer streaming nos seguintes cenários.

1. Se você já usa o AWS Glue Spark para processamento em lote, o AWS Glue Streaming é a escolha ideal para você. Ele fornece uma transição sem complicação para o desenvolvimento de trabalhos de streaming sem a necessidade de aprender uma nova linguagem ou estrutura. Aproveitando seu conhecimento e sua infraestrutura existentes, o AWS Glue Streaming simplifica o processo de desenvolvimento de tarefas e permite que você estenda facilmente seus recursos de processamento de dados para cenários de streaming em tempo real.
2. Se você precisa de um serviço ou produto unificado para lidar com cargas de trabalho em lote, streaming e orientadas por eventos, o AWS Glue streaming é a solução para você. Com o AWS Glue Streaming, você pode consolidar suas necessidades de processamento de dados em uma única estrutura, eliminando a complexidade do gerenciamento de vários sistemas. Isso possibilita o desenvolvimento e a manutenção eficientes de diversos fluxos de trabalho de dados, ao mesmo tempo em que garante consistência e compatibilidade entre diferentes tipos de workloads.
3. AWS Glue O streaming é adequado para cenários que envolvem volumes de dados de streaming extremamente grandes e transformações complexas, como junções entre fluxos ou bancos de dados relacionais. Ele pode processar e analisar fluxos massivos de dados com eficiência, possibilitando que você lide com workloads complexas com facilidade. Quer se trate de ingestão de dados em alta velocidade ou manipulações complexas de dados, a escalabilidade e os recursos avançados de processamento do AWS Glue Streaming garantem um desempenho ideal e resultados precisos.
4. Se você preferir uma abordagem visual para criar trabalhos de streaming, AWS Glue oferece o AWS Glue Studio, com o qual você pode projetar e gerenciar visualmente seus aplicativos de streaming, simplificando o processo de desenvolvimento. Essa interface intuitiva possibilita que os desenvolvedores criem, configurem e monitorem fluxos de trabalho de streaming usando uma interface visual, o que reduz a curva de aprendizado e aumenta a produtividade.

5. AWS Glue O streaming é uma excelente opção para casos de near-real-time uso em que há SLAs (contratos de nível de serviço) rigorosos por mais de 10 segundos.
6. Se você estiver criando um data lake transacional usando o Apache Iceberg, o Apache Hudi ou o Delta Lake, o AWS Glue Streaming fornece suporte nativo para esses formatos de tabela aberta. Essa integração sem complicações possibilita o processamento de dados de streaming diretamente desses data lakes transacionais, garantindo integridade, compatibilidade e consistência de dados.
7. Quando precisar ingerir dados de streaming para uma variedade de destinos de dados: o AWS Glue streaming fornece destinos nativos para uma variedade de destinos de dados, como Amazon Redshift, Amazon RDS, Amazon Aurora, Oracle, SQL Server e outros destinos.

Fonte de dados compatíveis

AWS Glue O streaming é compatível com as seguintes fontes de dados:

- Amazon Kinesis
- Amazon MSK (Managed Streaming for Apache Kafka)
- Apache Kafka autogerenciado

Destinos de dados com suporte

AWS Glue O streaming suporta uma variedade de destinos de dados, como:

- Destinos de dados suportados pelo AWS Glue Data Catalog
- Amazon S3
- Amazon Redshift
- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server
- Snowflake
- Qualquer banco de dados que possa ser conectado usando JDBC
- Apache Iceberg, Delta e Apache Hudi
- AWS Glue Conectores do Marketplace

Tutorial: desenvolvimento da sua primeira workload de streaming usando o AWS Glue Studio

Neste tutorial, você aprenderá como criar um trabalho de streaming usando o AWS Glue Studio. O AWS Glue Studio corresponde a uma interface visual para a criação de trabalhos do AWS Glue.

É possível criar trabalhos de extração, transformação e carregamento (ETL) de streaming que são executados continuamente e consomem dados de fontes de streaming no Amazon Kinesis Data Streams, no Apache Kafka e no Amazon Managed Streaming for Apache Kafka (Amazon MSK).

Pré-requisitos

Para seguir este tutorial, você precisará de um usuário com permissões do Console da AWS para usar o AWS Glue, o Amazon Kinesis, o Amazon S3, o Amazon Athena, o AWS CloudFormation, o AWS Lambda e o Amazon Cognito.

Consumo de dados de streaming do Amazon Kinesis

Tópicos

Geração de dados de simulação com o Kinesis Data Generator

Você pode gerar dados de amostra sinteticamente no formato JSON usando o Kinesis Data Generator (KDG). É possível encontrar instruções completas e detalhes na documentação da ferramenta.

1. Para começar, clique em

   para executar um modelo do AWS CloudFormation em seu ambiente da AWS.

   > **Note**
   >
   > Pode ocorrer uma falha no modelo do CloudFormation porque alguns recursos, como o usuário do Amazon Cognito para o Kinesis Data Generator, já existem em sua conta

da AWS. Isso pode acontecer porque você já configurou esses recursos seguindo outro tutorial ou outra publicação de blog. Para resolver isso, é possível experimentar o modelo em uma nova conta da AWS para começar do zero ou explorar uma região da AWS diferente. Essas opções permitem que você execute o tutorial sem entrar em conflito com os recursos existentes.

O modelo provisiona um fluxo de dados do Kinesis e uma conta do Kinesis Data Generator para você. Além disso, ele cria um bucket do Amazon S3 para armazenar os dados e um perfil de serviço do Glue com a permissão necessária para a execução deste tutorial.

2. Digite um Nome de usuário e uma Senha que o KDG usará para realizar a autenticação. Anote o nome de usuário e a senha para uso posterior.
3. Selecione Próximo até a última etapa. Reconheça a criação de recursos do IAM. Verifique se há erros na parte superior da tela, como a senha não atender aos requisitos mínimos, e implante o modelo.
4. Navegue até a guia Saídas da pilha. Depois que o modelo for implantado, ele exibirá a propriedade gerada KinesisDataGeneratorUrl. Clique nesse URL.
5. Digite o Nome de usuário e a Senha que você anotou.
6. Selecione a região que você está usando e selecione o fluxo do Kinesis `GlueStreamTest-{AWS::AccountId}`.
7. Insira o seguinte modelo:

```
{
    "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
        }
    )}},
```

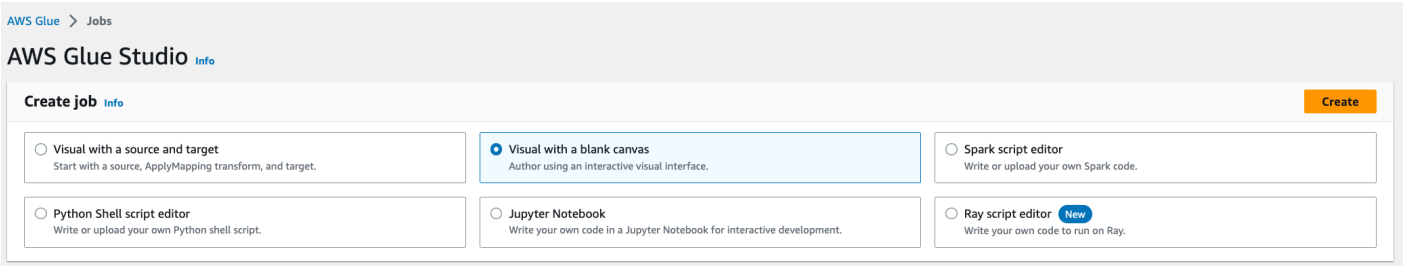
```
    "minutevolume": {{random.number(
        {
            "min":5,
            "max":8
        }
    )}},
    "manufacturer": "{{random.arrayElement(
        ["3M", "GE","Vyaire", "Getinge"]
    )}}"
}
```

Agora, é possível visualizar dados de simulação com Testar modelo e ingerir os dados de simulação no Kinesis com Enviar dados.

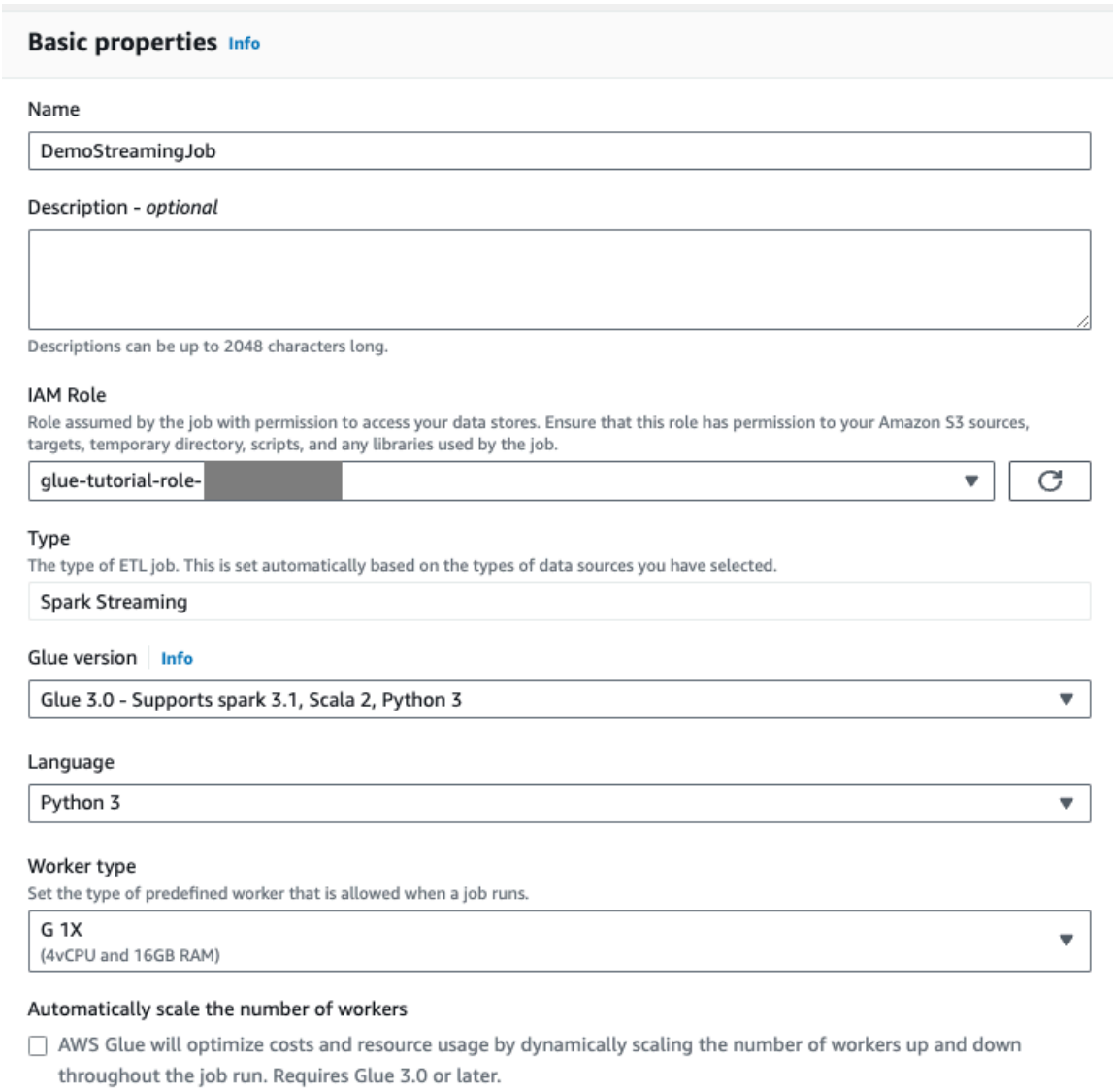
8. Clique em Enviar dados e gere de cinco a dez mil registros para o Kinesis.

Criação de um trabalho de streaming do AWS Glue com o AWS Glue Studio

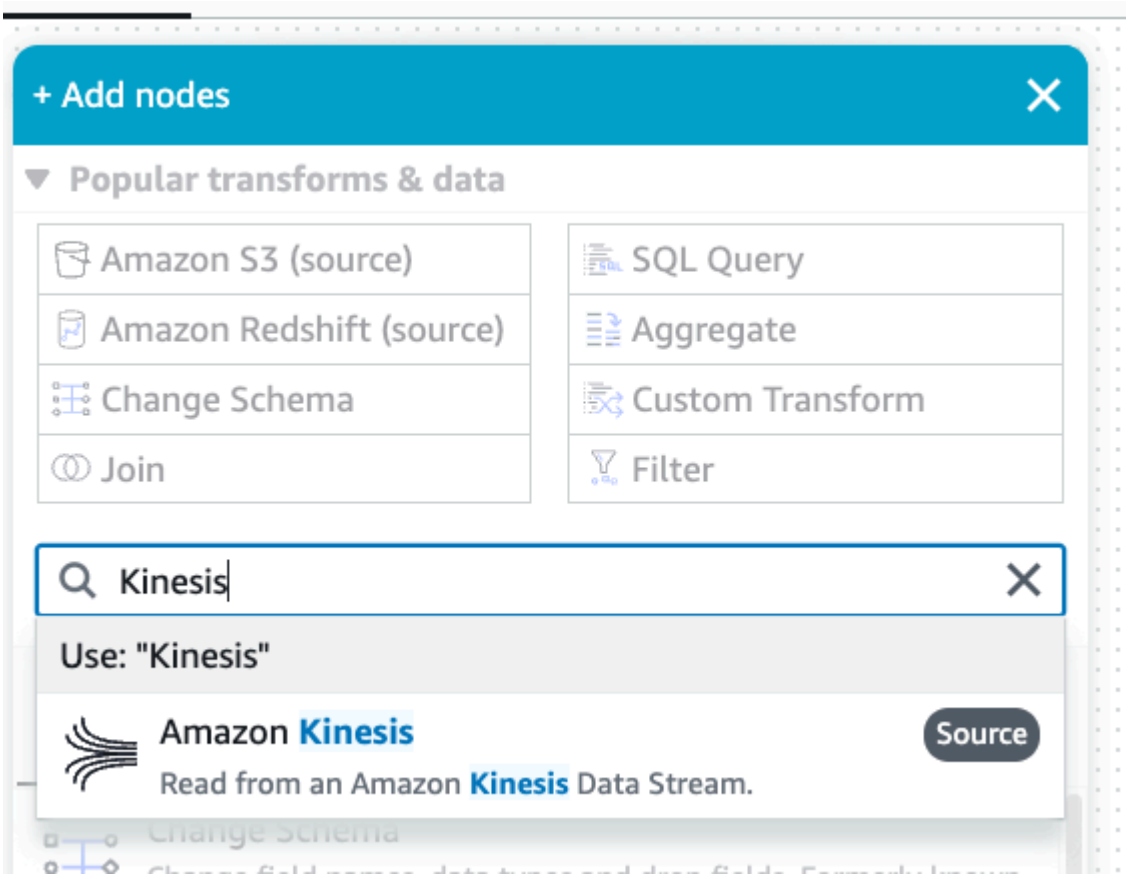
1. Navegue até AWS Glue no console da mesma região.
2. Selecione Trabalhos de ETL na barra de navegação do lado esquerdo em Integração de dados e ETL.
3. Crie um trabalho do AWS Glue usando Elemento visual com uma tela em branco.



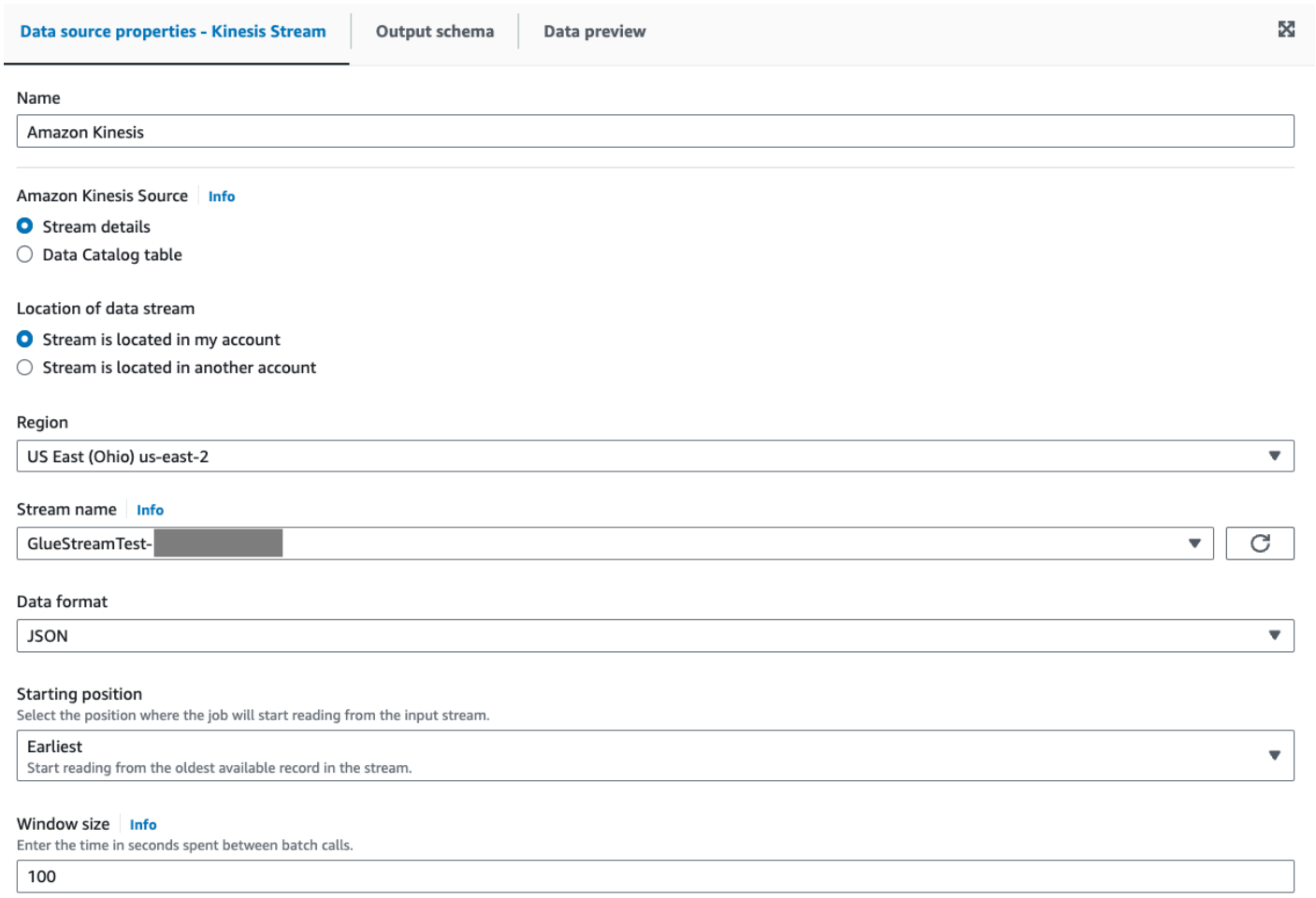
4. Navegue até a guia Detalhes do trabalho.
5. Para o nome do trabalho do AWS Glue, insira `DemoStreamingJob`.
6. Para o Perfil do IAM, selecione o perfil provisionado pelo modelo do CloudFormation, `glue-tutorial-role-${AWS::AccountId}`.
7. Para a Versão do Glue, selecione Glue 3.0. Deixe todas as outras opções conforme o padrão.



8. Navegue até a guia Elemento visual.
9. Clique no ícone de adição. Digite Kinesis na barra de pesquisa. Selecione a fonte de dados do Amazon Kinesis.



10. Selecione Detalhes do fluxo para a Fonte do Amazon Kinesis na guia Propriedades da fonte de dados: fluxo do Kinesis.
11. Selecione O fluxo está localizado em minha conta para Localização do fluxo de dados.
12. Selecione a região que você está usando.
13. Selecione o fluxo `GlueStreamTest-{AWS::AccountId}`.
14. Mantenha todas as outras configurações conforme o padrão.

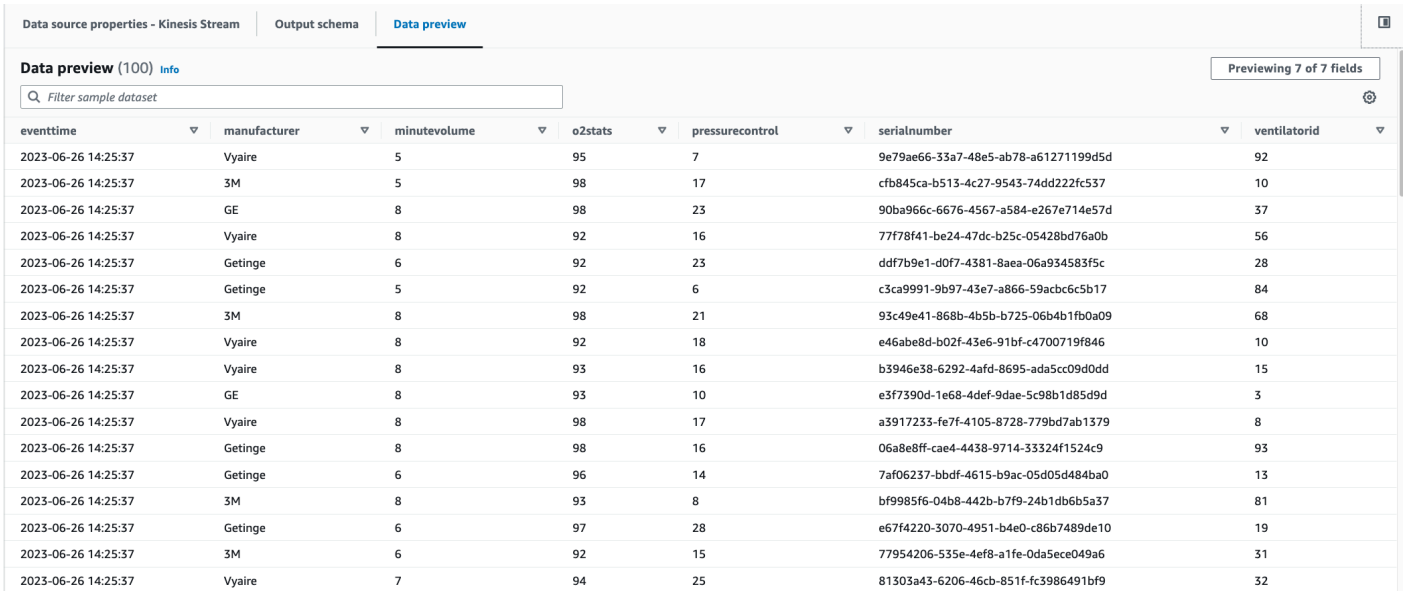


15. Navegue até a guia Pré-visualização de dados.

16. Clique em Iniciar sessão de pré-visualização de dados, que visualiza previamente os dados de simulação gerados pelo KDG. Escolha o perfil de serviço do Glue que você criou anteriormente para o trabalho do AWS Glue Streaming.

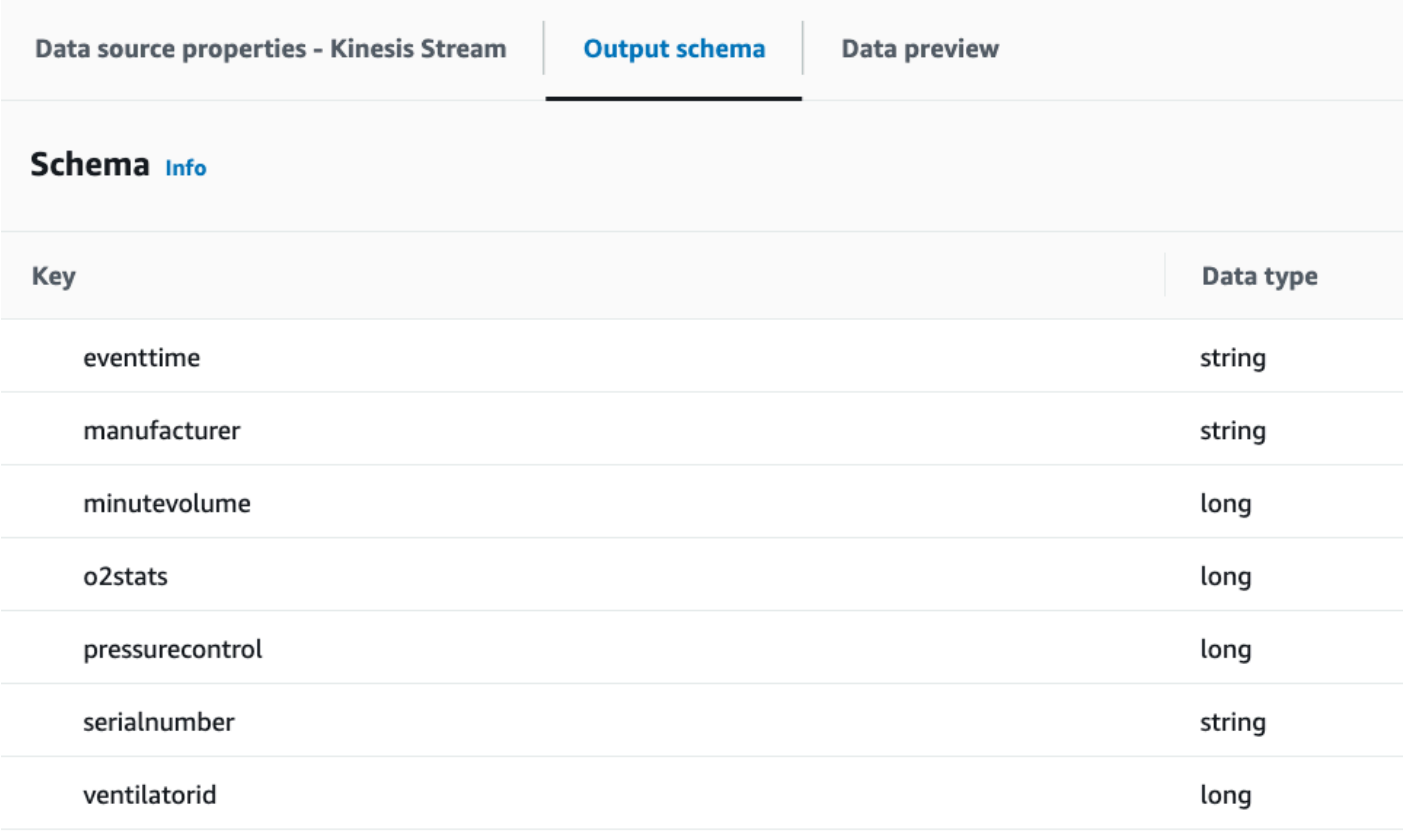
    Demora de 30 a 60 segundos para que os dados de pré-visualização sejam mostrados. Se aparecer Não há dados a serem exibidos, clique no ícone de engrenagem e altere o Número de linhas para amostra para `100`.

    Você pode visualizar os dados de amostra, como apresentado abaixo:



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

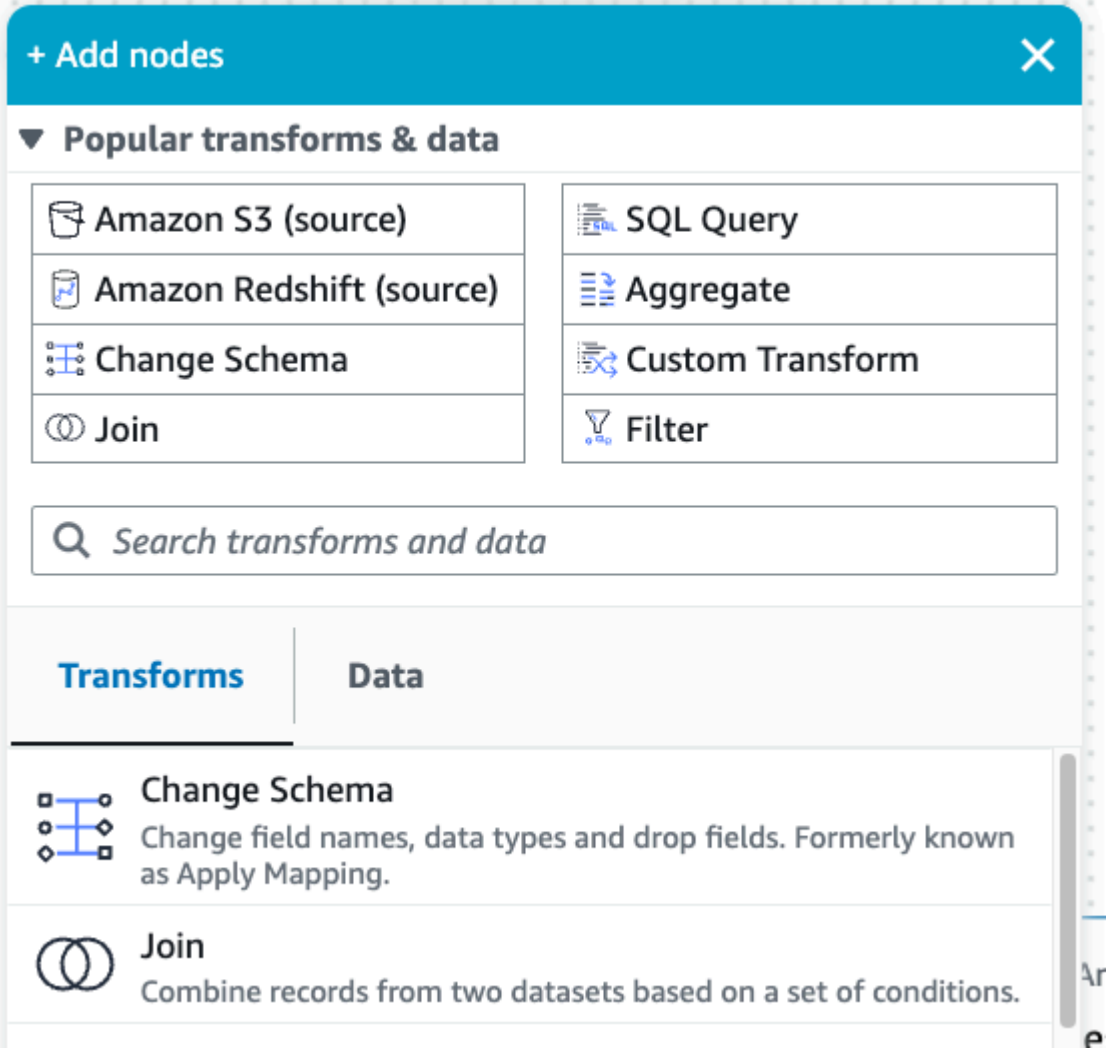
Além disso, é possível visualizar o esquema inferido na guia Esquema de saída.



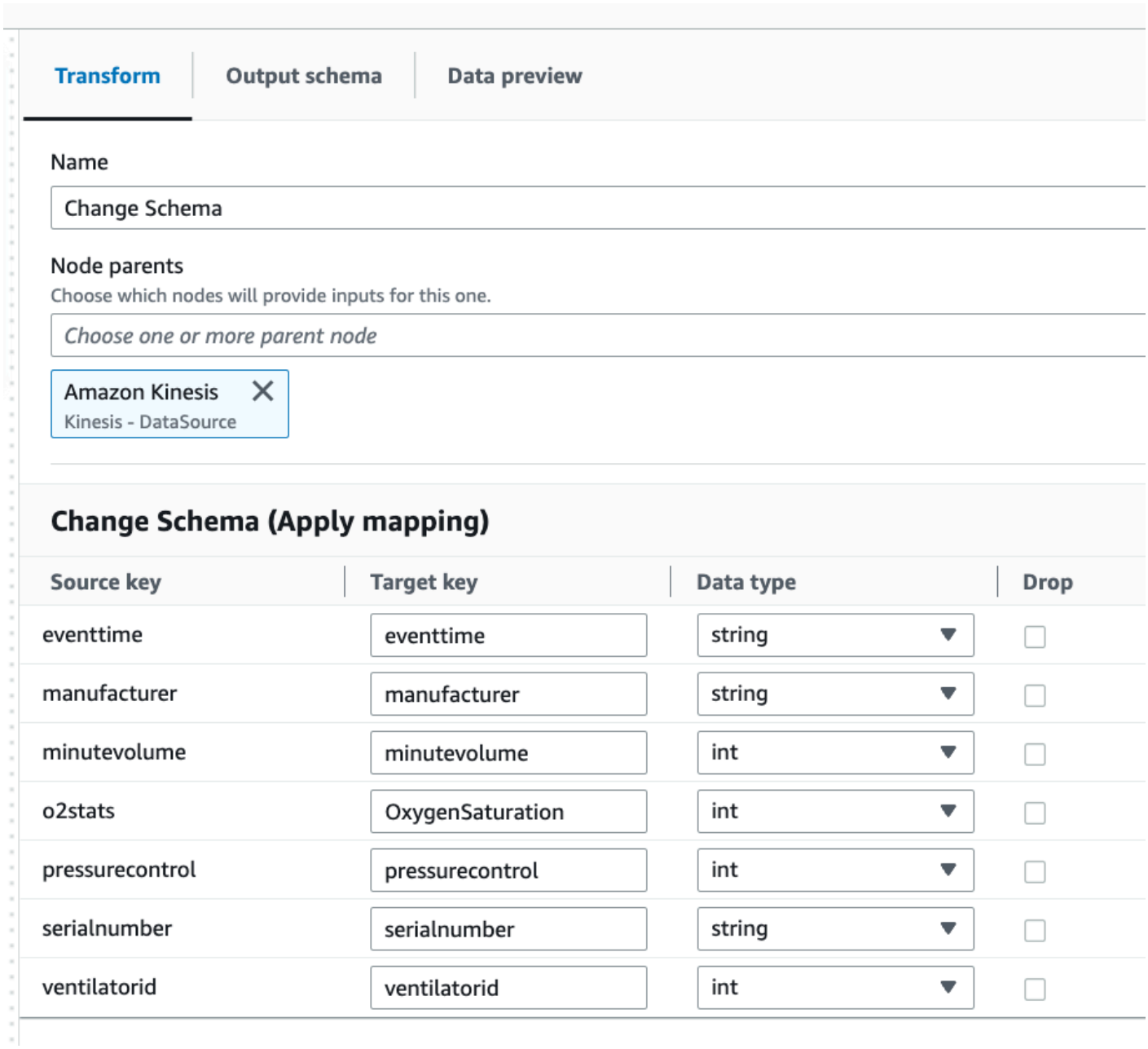
Key	Data type
eventtime	string
manufacturer	string
minutevolume	long
o2stats	long
pressurecontrol	long
serialnumber	string
ventilatorid	long

Execução de uma transformação e armazenamento do resultado transformado no Amazon S3

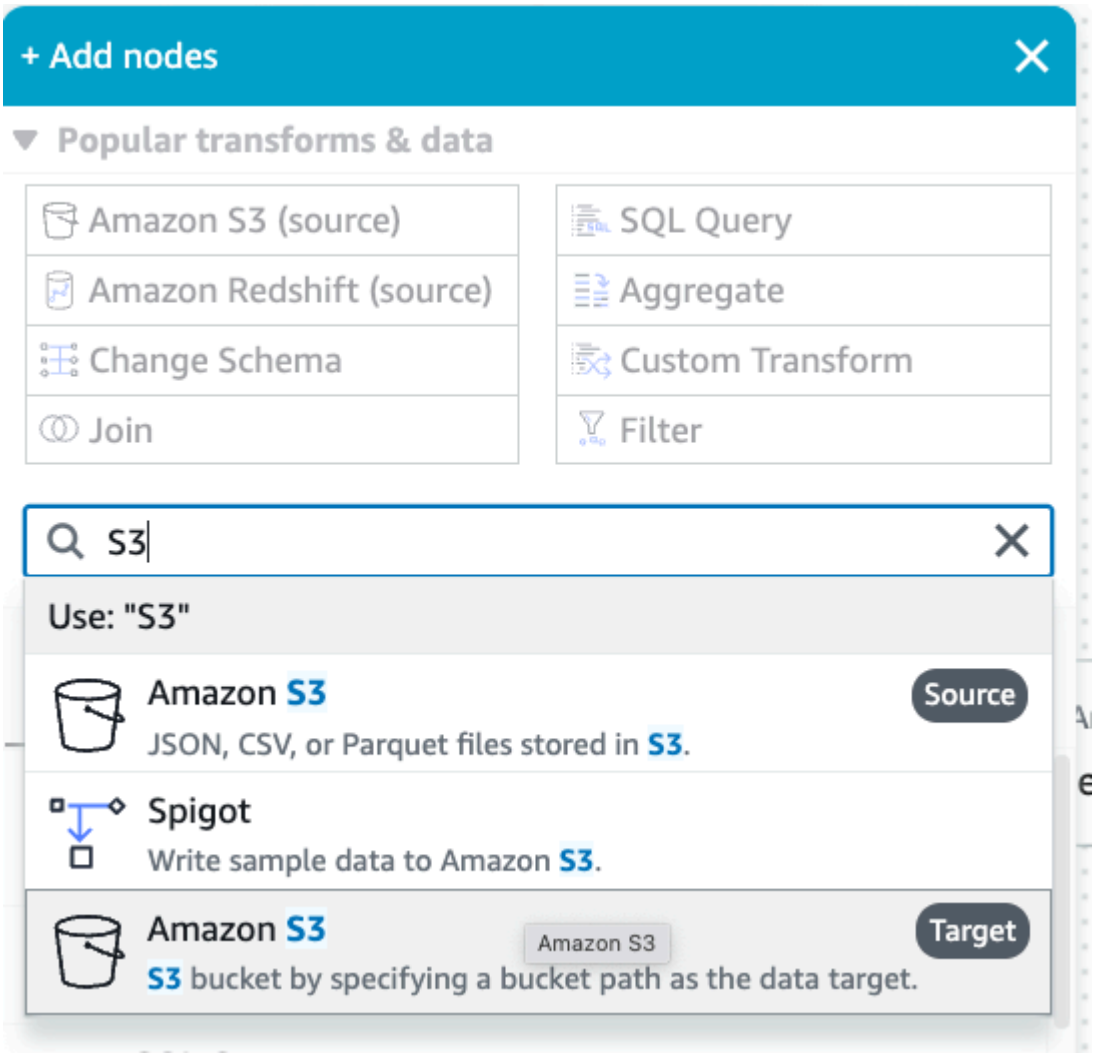
1. Com o nó de origem selecionado, clique no ícone de adição no canto superior esquerdo para adicionar uma etapa de Transformação.
2. Selecione a etapa Alterar esquema.



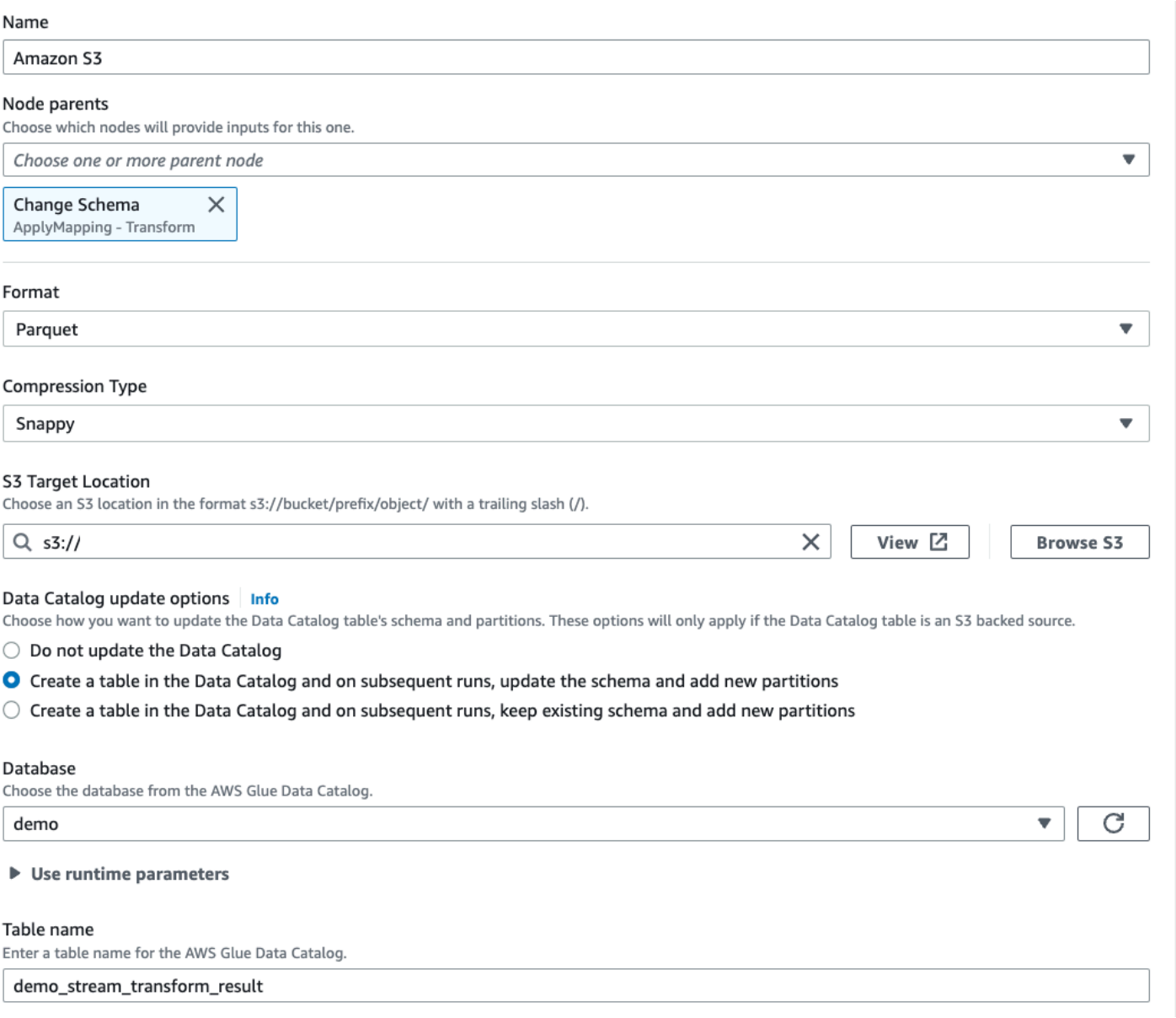
3. É possível renomear campos e converter o tipo de dados dos campos nesta etapa. Renomeie a coluna `o2stats` para `OxygenSaturation` e converta todos os tipos de dados `long` para `int`.



4. Clique no ícone de adição para adicionar um destino do Amazon S3. Insira S3 na caixa de pesquisa e selecione a etapa de transformação Amazon S3: destino.

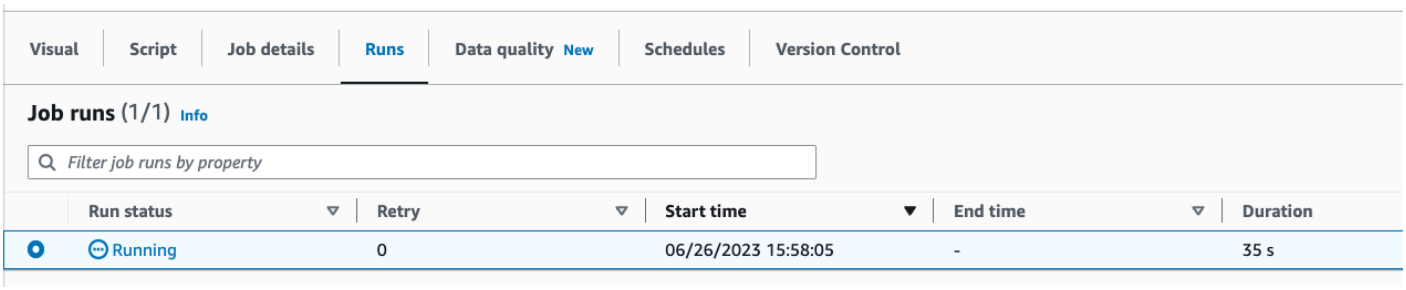


5. Selecione Parquet como o formato do arquivo de destino.
6. Selecione Snappy como o tipo de compactação.
7. Insira um Local de destino do S3 criado pelo modelo do CloudFormation, `streaming-tutorial-s3-target-{AWS::AccountId}`.
8. Selecione Criar uma tabela no Catálogo de Dados e, em execuções subsequentes, atualizar o esquema e adicionar novas partições.
9. Insira o nome do Banco de dados e da Tabela de destino para armazenar o esquema da tabela de destino do Amazon S3.

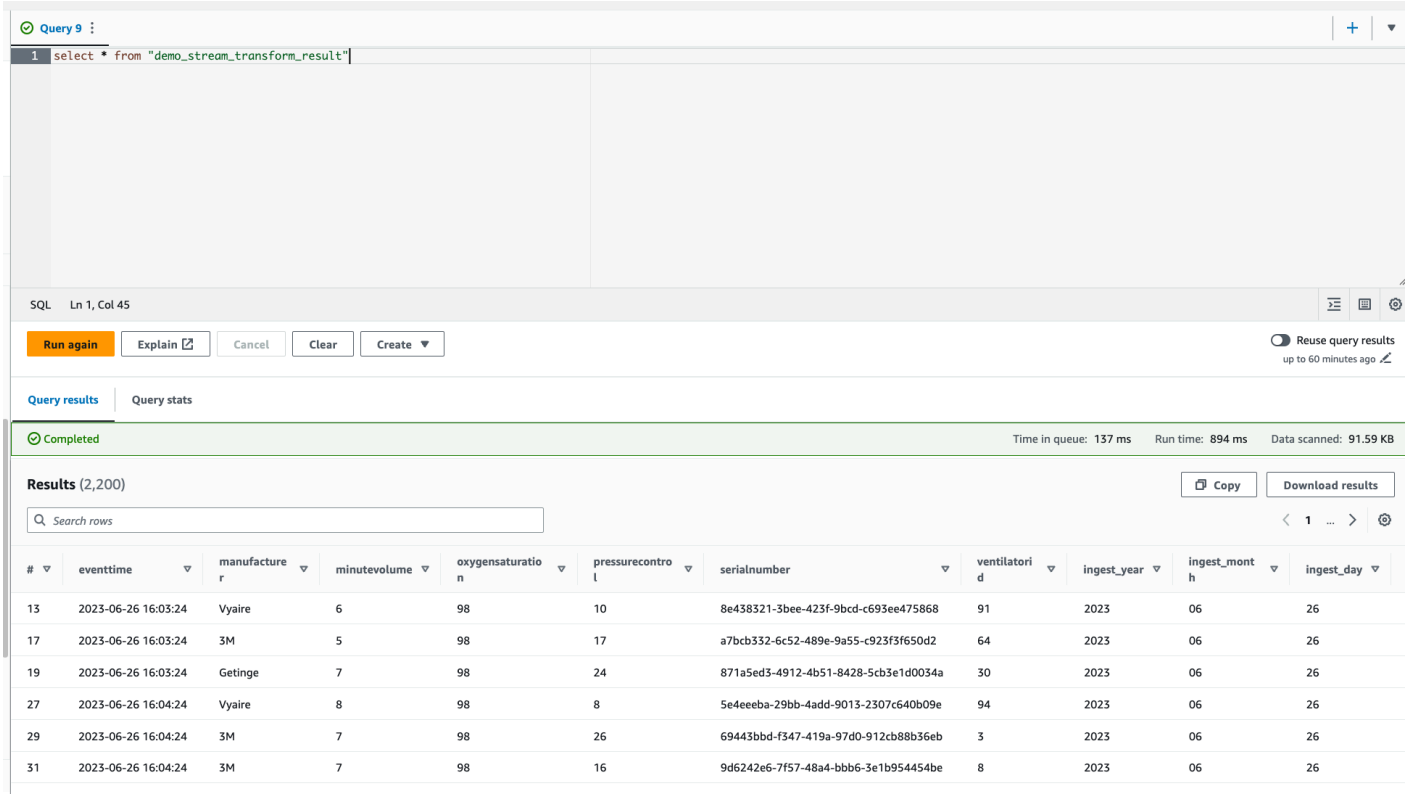


10. Clique na guia Script para visualizar o código gerado.
11. Clique em Salvar no canto superior direito para salvar o código de ETL e, em seguida, clique em Executar para iniciar o trabalho de streaming do AWS Glue.

    É possível encontrar o Status de execução na guia Execuções. Deixe o trabalho ser executado entre três e cinco minutos e, em seguida, interrompa-o.



12.Verifique a nova tabela criada no Amazon Athena.



Tutorial: desenvolvimento da sua primeira workload de streaming usando cadernos do AWS Glue Studio

Neste tutorial, você explorará como aproveitar os cadernos do AWS Glue Studio para desenvolver e otimizar, de forma interativa, os trabalhos de ETL para o processamento de dados quase em tempo real. Quer essa seja sua primeira experiência com o AWS Glue ou você esteja procurando aprimorar seu conjunto de habilidades, este guia orientará você durante o processo, possibilitando o aproveitamento de todo o potencial dos cadernos de sessão interativa do AWS Glue.

Com o AWS Glue Streaming, é possível criar trabalhos de extração, transformação e carregamento (ETL) de streaming que são executados continuamente e consomem dados de fontes de streaming, como o Amazon Kinesis Data Streams, o Apache Kafka e o Amazon Managed Streaming for Apache Kafka (Amazon MSK).

Pré-requisitos

Para seguir este tutorial, você precisará de um usuário com permissões do Console da AWS para usar o AWS Glue, o Amazon Kinesis, o Amazon S3, o Amazon Athena, o AWS CloudFormation, o AWS Lambda e o Amazon Cognito.

Consumo de dados de streaming do Amazon Kinesis

Tópicos

Geração de dados de simulação com o Kinesis Data Generator

> **Note**
>
> Se você já concluiu a etapa anterior apresentada em Tutorial: desenvolvimento da sua primeira workload de streaming usando o AWS Glue Studio, já tem o Kinesis Data Generator instalado em sua conta e pode pular as etapas de 1 a 8 abaixo e avançar para a seção Criação de um trabalho de streaming do AWS Glue com o AWS Glue Studio.

Você pode gerar dados de amostra sinteticamente no formato JSON usando o Kinesis Data Generator (KDG). É possível encontrar instruções completas e detalhes na documentação da ferramenta.

1. Para começar, clique em

   para executar um modelo do AWS CloudFormation em seu ambiente da AWS.

> **Note**
>
> Pode ocorrer uma falha no modelo do CloudFormation porque alguns recursos, como o usuário do Amazon Cognito para o Kinesis Data Generator, já existem em sua conta da AWS. Isso pode acontecer porque você já configurou esses recursos seguindo outro tutorial ou outra publicação de blog. Para resolver isso, é possível experimentar o modelo em uma nova conta da AWS para começar do zero ou explorar uma região da AWS diferente. Essas opções permitem que você execute o tutorial sem entrar em conflito com os recursos existentes.

O modelo provisiona um fluxo de dados do Kinesis e uma conta do Kinesis Data Generator para você.

2. Digite um Nome de usuário e uma Senha que o KDG usará para realizar a autenticação. Anote o nome de usuário e a senha para uso posterior.
3. Selecione Próximo até a última etapa. Reconheça a criação de recursos do IAM. Verifique se há erros na parte superior da tela, como a senha não atender aos requisitos mínimos, e implante o modelo.
4. Navegue até a guia Saídas da pilha. Depois que o modelo for implantado, ele exibirá a propriedade gerada KinesisDataGeneratorUrl. Clique nesse URL.
5. Digite o Nome de usuário e a Senha que você anotou.
6. Selecione a região que você está usando e selecione o fluxo do Kinesis `GlueStreamTest-{AWS::AccountId}`.
7. Insira o seguinte modelo:

```
{
    "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"]
    )}}"
}
```

Agora, é possível visualizar dados de simulação com Testar modelo e ingerir os dados de simulação no Kinesis com Enviar dados.

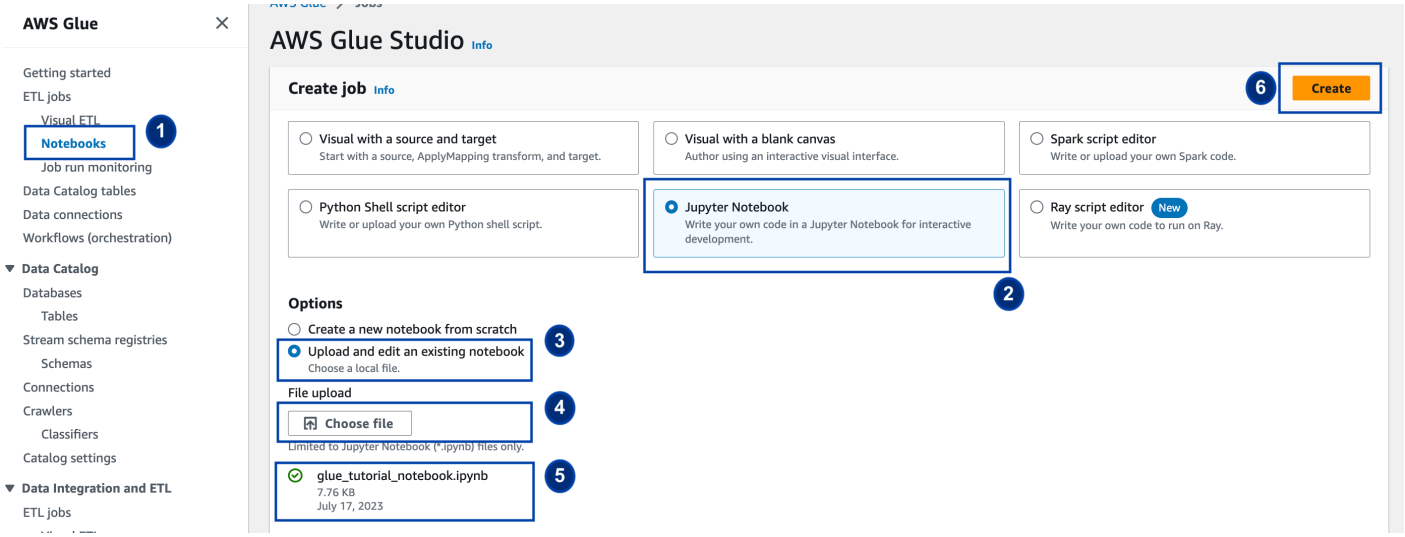
8. Clique em Enviar dados e gere de cinco a dez mil registros para o Kinesis.

Criação de um trabalho de streaming do AWS Glue com o AWS Glue Studio

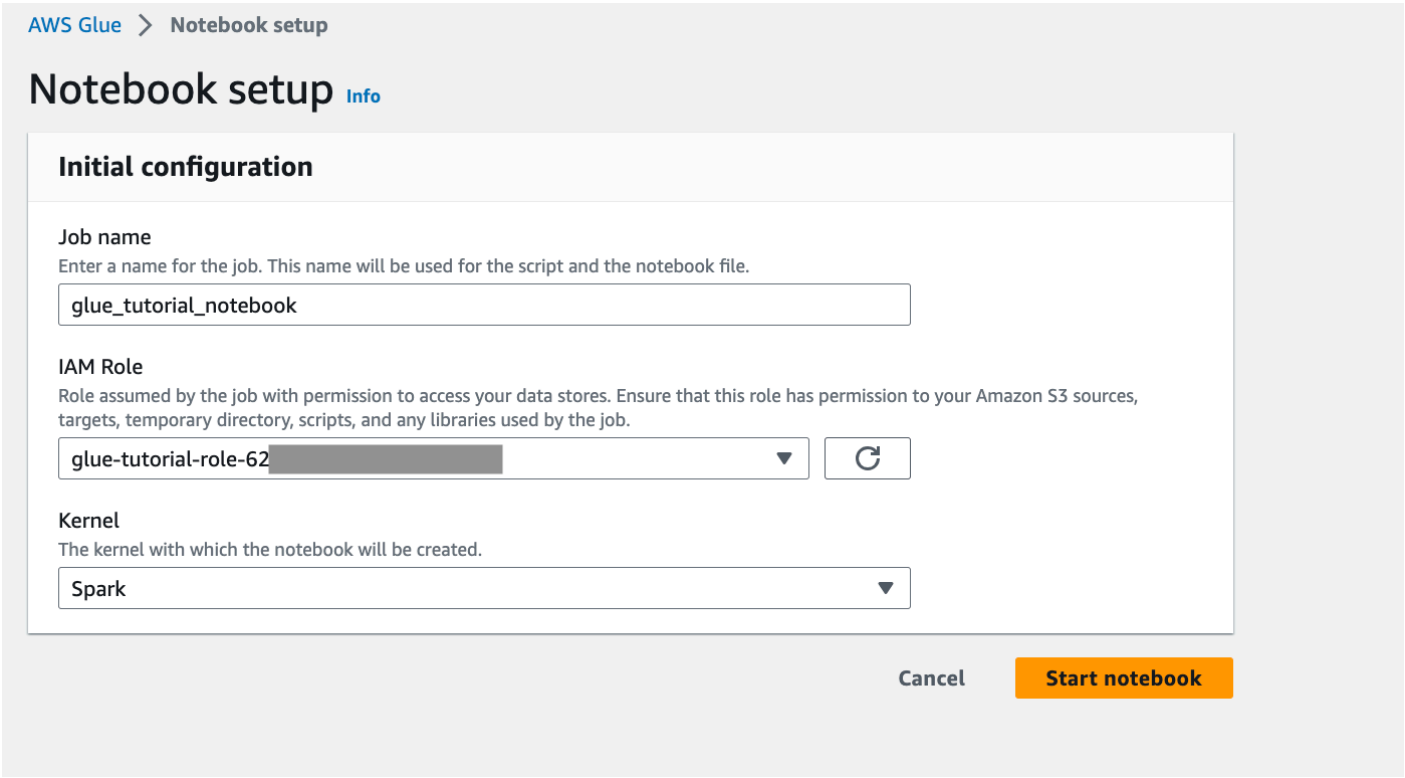
O AWS Glue Studio corresponde a uma interface visual que simplifica o processo de criação, orquestração e monitoramento de pipelines de integração de dados. Ele possibilita que os usuários desenvolvam pipelines de transformação de dados sem a necessidade de escrever códigos extensos. Além da experiência de criação de trabalhos visuais, o AWS Glue Studio também inclui um caderno Jupyter apoiado por sessões interativas do AWS Glue, que você usará no restante deste tutorial.

Configuração do trabalho de sessões interativas do AWS Glue Streaming

1. Faça download do arquivo do caderno fornecido e salve-o em um diretório local.
2. Abra o console do AWS Glue e, no painel esquerdo, clique em Cadernos > Caderno Jupyter > Fazer upload e editar um caderno existente. Faça upload do caderno usando a etapa anterior e clique em Criar.



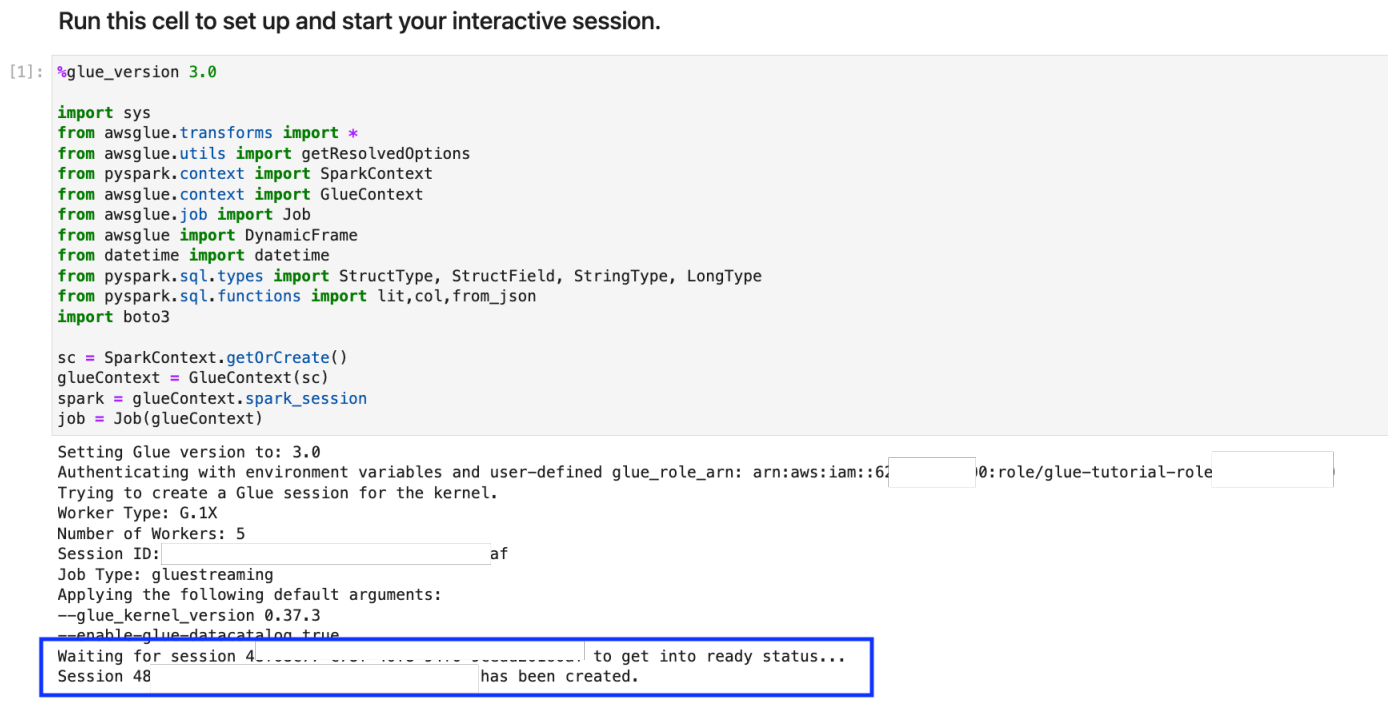
3. Forneça um nome e um perfil ao trabalho e selecione o kernel padrão do Spark. Em seguida, clique em Iniciar caderno. Para o Perfil do IAM, selecione o perfil provisionado pelo modelo do CloudFormation. É possível visualizar isso na guia Saídas do CloudFormation.



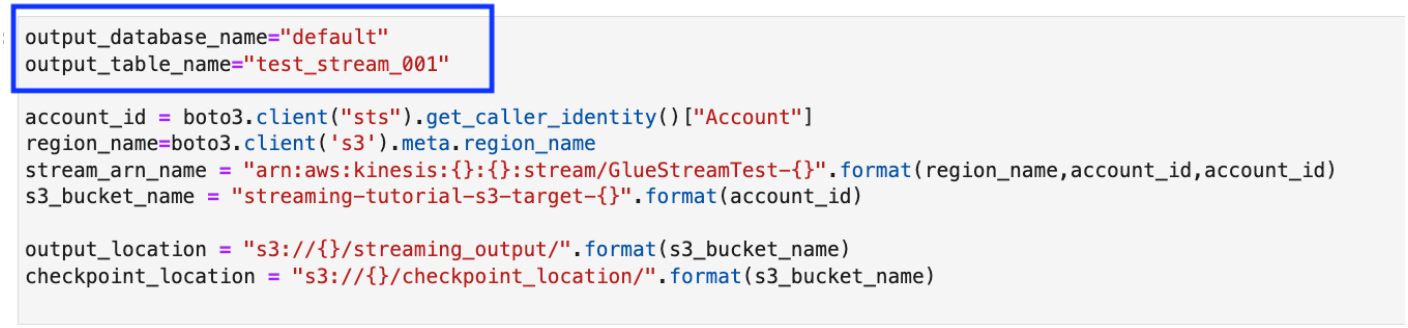
O caderno contém todas as instruções necessárias para continuar o tutorial. É possível executar as instruções no caderno ou seguir este tutorial para continuar com o desenvolvimento do trabalho.

Execução das células do caderno

1. (Opcional) A primeira célula de código, `%help`, lista todas as mágicas disponíveis para o caderno. É possível pular essa célula por enquanto, mas fique à vontade para explorá-la.
2. Comece com o próximo bloco de código, `%streaming`. Essa mágica define o tipo de trabalho para o streaming, o que permite desenvolver, depurar e implantar um trabalho de ETL de streaming do AWS Glue.
3. Execute a próxima célula para criar uma sessão interativa do AWS Glue. A célula de saída tem uma mensagem que confirma a criação da sessão.



4. A próxima célula define as variáveis. Substitua os valores pelos valores apropriados para o seu trabalho e execute a célula. Por exemplo:



5. Como os dados já estão sendo transmitidos para o Kinesis Data Streams, a próxima célula consumirá os resultados do fluxo. Execute a próxima célula. Como não há instruções de impressão, não existe uma saída esperada para essa célula.

6. Na célula apresentada a seguir, você explora o fluxo de entrada ao utilizar um conjunto de amostra e imprimir o esquema e os dados reais dele. Por exemplo:

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 seguir, defina a lógica real de transformação de dados. A célula consiste no método `processBatch`, que é acionado durante cada micro lote. Execute a célula. Em um nível superior, fazemos o seguinte com o fluxo de entrada:

   a. Selecione um subconjunto das colunas de entrada.

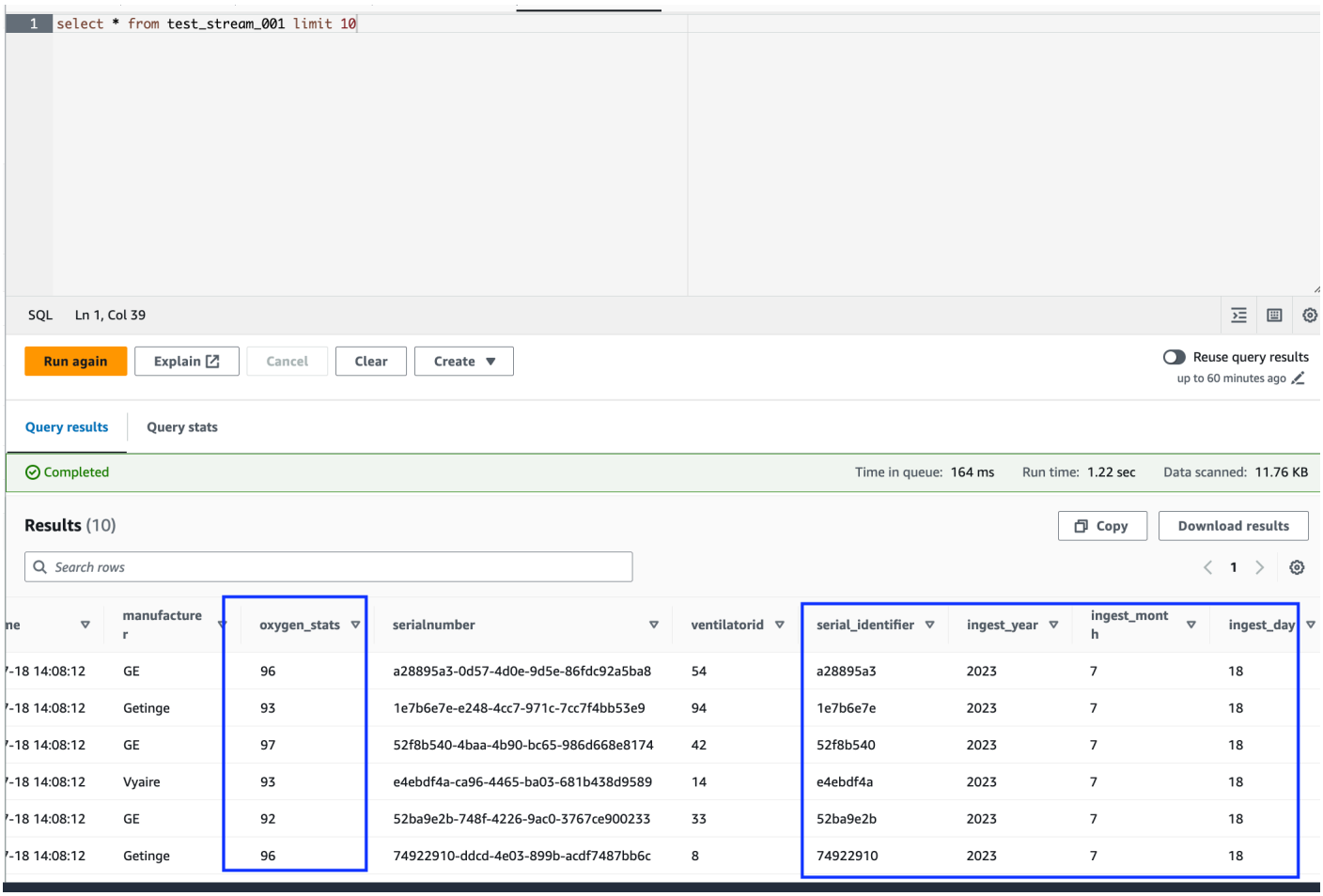
   b. Renomeie uma coluna (o2stats para oxygen_stats).

   c. Derive novas colunas (serial_identifier, ingest_year, ingest_month e ingest_day).

   d. Armazene os resultados em um bucket do Amazon S3 e também crie uma tabela do catálogo particionada do AWS Glue.

8. Na última célula, você aciona o lote do processo a cada dez segundos. Execute a célula e aguarde cerca de 30 segundos para que ela preencha o bucket do Amazon S3 e a tabela de catálogo do AWS Glue.

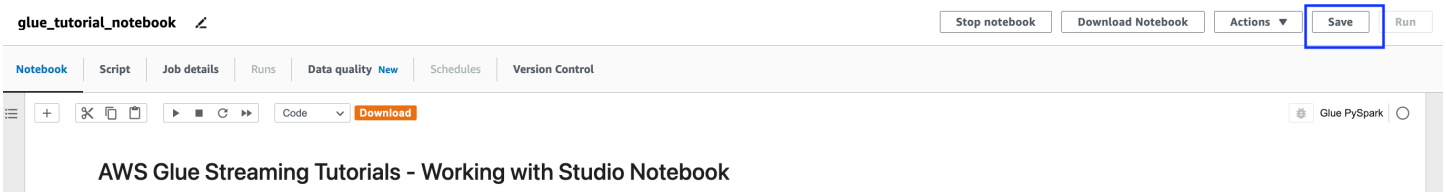
9. Por fim, navegue pelos dados armazenados usando o editor de consultas do Amazon Athena. É possível visualizar a coluna renomeada e também as novas partições.



O caderno contém todas as instruções necessárias para continuar o tutorial. É possível executar as instruções no caderno ou seguir este tutorial para continuar com o desenvolvimento do trabalho.

Salvamento e execução do trabalho do AWS Glue

Com o desenvolvimento e o teste da aplicação concluídos usando o caderno de sessões interativas, clique em Salvar na parte superior da interface do caderno. Depois de salva, também é possível executar a aplicação como um trabalho.



Limpeza

Para evitar cobranças adicionais em sua conta, interrompa o trabalho de streaming que você iniciou como parte das instruções. É possível fazer isso ao interromper o caderno, o que encerrará a sessão. Esvazie o bucket do Amazon S3 e exclua a pilha do AWS CloudFormation provisionada anteriormente.

Conclusão

Neste tutorial, demonstramos como fazer o seguinte usando o caderno do AWS Glue Studio:

- Criar um trabalho de ETL de streaming usando cadernos;
- Pré-visualizar os fluxos de dados de entrada;
- Codificar e corrigir problemas sem a necessidade de publicar trabalhos do AWS Glue;
- Analisar o código de trabalho de ponta a ponta, remover qualquer depuração e imprimir instruções ou células do caderno;
- Publicar o código como um trabalho do AWS Glue.

O objetivo deste tutorial é proporcionar a você uma experiência prática de trabalho com o AWS Glue Streaming e com as sessões interativas. Recomendamos que você o use como referência para seus casos de uso individuais do AWS Glue Streaming. Para ter mais informações, consulte Conceitos básicos das sessões interativas do AWS Glue.

Conceitos do AWS Glue Streaming

As seções apresentadas a seguir fornecem informações sobre os conceitos do AWS Glue Streaming.

Tópicos

Anatomia de um trabalho de streaming do AWS Glue

Os trabalhos de streaming do AWS Glue operam no paradigma de streaming do Spark e aproveitam o streaming estruturado da estrutura do Spark. Os trabalhos de streaming pesquisam constantemente a fonte de dados de streaming, em um intervalo de tempo específico, para buscar registros como micro lotes. As seções apresentadas a seguir examinam as diferentes partes de um trabalho de streaming do 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

O método `forEachBatch` é o ponto de entrada de uma execução de trabalho de streaming do AWS Glue. Os trabalhos de streaming do AWS Glue usam o método `forEachBatch` para pesquisar dados. Ele funciona como um iterador que permanece ativo durante o ciclo de vida do trabalho de streaming e pesquisa regularmente a fonte de streaming em busca de novos dados, além de processar os dados mais recentes em micro lotes.

```
glueContext.forEachBatch(
    frame=dataFrame_AmazonKinesis_node1696872487972,
    batch_function=processBatch,
```

```
    options={
        "windowSize": "100 seconds",
        "checkpointLocation": args["TempDir"] + "/" + args["JOB_NAME"] + "/
checkpoint/",
    },
)
```

Configure a propriedade `frame` de `forEachBatch` para especificar uma fonte de streaming. Neste exemplo, o nó de origem que você criou na tela em branco durante a criação do trabalho é preenchido com o DataFrame padrão do trabalho. Configure a propriedade `batch_function` como a `function` que você decide invocar para cada operação de micro lote. Você deve definir uma função para tratar da transformação em lote relacionada aos dados de entrada.

Origem

Na primeira etapa da função `processBatch`, o programa verifica a contagem de registros do DataFrame que você definiu como a propriedade frame de `forEachBatch`. O programa acrescenta um carimbo de data/hora de ingestão a um DataFrame que não está vazio. A cláusula `data_frame.count()>0` determina se o micro lote mais recente não está vazio e se está pronto para o processamento adicional.

```
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",
        )
```

Mapeamento

A próxima seção do programa corresponde à aplicação do mapeamento. O método `Mapping.apply` em um DataFrame do Spark permite definir regras de transformação em torno de elementos de dados. Normalmente, é possível renomear, alterar o tipo de dados ou aplicar uma função personalizada na coluna de dados da fonte e mapeá-los para as colunas 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",
    )
        )
```

Sink

Nesta seção, o conjunto de dados de entrada da fonte de streaming é armazenado em um local de destino. Neste exemplo, gravaremos os dados em um local do Amazon S3. Os detalhes da propriedade `AmazonS3_node_path` são preenchidos previamente, conforme determinado pelas configurações usadas durante a criação do trabalho na tela em branco. É possível definir `updateBehavior` com base em seu caso de uso e decidir Não atualizar a tabela do catálogo de dados ou Criar um catálogo de dados e atualizar o esquema do catálogo de dados em execuções subsequentes, ou criar uma tabela do catálogo e não atualizar a definição do esquema em execuções subsequentes.

A propriedade `partitionKeys` define a opção de partição de armazenamento. O comportamento padrão é particionar os dados de acordo com o `ingestion_time_columns` que foi disponibilizado na seção da fonte. A propriedade `compression` permite definir o algoritmo de compactação a ser aplicado durante a gravação de destino. Você tem as opções Snappy, LZO ou GZIP para definir uma técnica de compactação. A propriedade `enableUpdateCatalog` controla se a tabela do catálogo do AWS Glue precisa ser atualizada. As opções disponíveis para essa propriedade são `True` ou `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",
    )
```

Coletor do catálogo do AWS Glue

Esta seção do trabalho controla o comportamento de atualização da tabela do catálogo do AWS Glue. Defina as propriedades `catalogDatabase` and `catalogTableName` de acordo com o nome do banco de dados do Catálogo do AWS Glue e o nome da tabela associada ao trabalho do AWS Glue que você está projetando. É possível definir o formato do arquivo dos dados de destino por meio da propriedade `setFormat`. Neste exemplo, armazenaremos os dados no formato Parquet.

Depois de configurar e executar o trabalho de streaming do AWS Glue referente a este tutorial, os dados de streaming produzidos no Amazon Kinesis Data Streams serão armazenados no local do Amazon S3 em formato Parquet com compactação rápida. Em execuções com êxito do trabalho de streaming, será possível consultar os dados por meio do Amazon Athena.

```
    AmazonS3_node1696872743449 = setCatalogInfo(
        catalogDatabase =  "demo", catalogTableName = "demo_stream_transform_result"
    )
    AmazonS3_node1696872743449.setFormat("glueparquet")
    AmazonS3_node1696872743449.writeFormat("ChangeSchema_node16986872679326")
    )
```

Conexões do Kafka

Designa uma conexão com um cluster do Kafka ou um cluster do Amazon Managed Streaming for Apache Kafka.

É possível ler e gravar fluxos de dados do Kafka usando informações armazenadas em uma tabela do Catálogo de Dados ou fornecendo informações para acessar diretamente o fluxo de dados.

Você pode ler informações de Kafka em um Spark e depois DataFrame convertê-las em um Glue. AWS DynamicFrame Você pode DynamicFrames escrever no Kafka em um formato JSON. Se você acessar diretamente o fluxo de dados, use essas opções para fornecer as informações sobre como acessar o fluxo de dados.

Se você usar `getCatalogSource` ou `create_data_frame_from_catalog` para consumir registros de uma fonte de streaming do Kafka, ou `getCatalogSink` ou `write_dynamic_frame_from_catalog` para gravar registros no Kafka, e o trabalho tiver o banco de dados do catálogo de dados e as informações de nome da tabela, e poderá usá-los para obter alguns parâmetros básicos para leitura da fonte de streaming do Kafka. Se você usar `getSource`, `getCatalogSink`, `getSourceWithFormat`, `getSinkWithFormat`, `createDataFrameFromOptions`, `create_data_frame_from_options` ou `write_dynamic_frame_from_catalog`, será necessário especificar esses parâmetros básicos usando as opções de conexão descritas aqui.

Você pode especificar as opções de conexão para o Kafka usando os seguintes argumentos para os métodos especificados na classe `GlueContext`.

- Scala
  - `connectionOptions`: usar com `getSource`, `createDataFrameFromOptions`, `getSink`
  - `additionalOptions`: usar com `getCatalogSource`, `getCatalogSink`
  - `options`: usar com `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: usar com `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: usar com `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: usar com `getSource`, `getSink`

Para notas e restrições sobre trabalhos de ETL de streaming, consulte the section called “Notas e restrições sobre ETL de transmissão”.

Configurar o Kafka

Não há AWS pré-requisitos para se conectar aos streams do Kafka disponíveis pela Internet.

Você pode criar uma conexão AWS Glue Kafka para gerenciar suas credenciais de conexão. Para ter mais informações, consulte the section called “Criar uma conexão para um fluxo de dados do Kafka”. Na configuração do trabalho do AWS Glue, forneça *connectionName* como uma conexão de rede adicional e, em seguida, na chamada do método, forneça *connectionName* ao parâmetro. `connectionName`

Em certos casos, você precisará configurar pré-requisitos adicionais:

- Se estiver usando o Amazon Managed Streaming for Apache Kafka com autenticação do IAM, você precisará da configuração apropriada do IAM.
- Se estiver usando o Amazon Managed Streaming for Apache Kafka com uma Amazon VPC, você precisará da configuração apropriada do Amazon VPC. Você precisará criar uma conexão AWS Glue que forneça informações de conexão com a Amazon VPC. Você precisará que sua configuração de trabalho inclua a conexão AWS Glue como uma conexão de rede adicional.

Para obter mais informações sobre pré-requisitos de trabalho de ETL de streaming, consulte the section called “Trabalhos de transmissão de ETL”.

Exemplo: leitura de fluxos do Kafka

Usado em conjunto com the section called “forEachBatch”.

Exemplo para fonte de transmissão do 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)
```

Exemplo: gravação em fluxos do Kafka

Exemplos de gravação no Kafka:

Exemplo com o 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()
```

Exemplo com o 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)
```

Referência de opções de conexão do Kafka

Ao ler, use as seguintes opções de conexão com `"connectionType": "kafka"`:

- `"bootstrap.servers"` (obrigatório) uma lista de URLs do servidor de bootstrap, por exemplo, como `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`. Essa opção deve ser especificada na chamada de API ou definida nos metadados da tabela no Data Catalog.
- `"security.protocol"` (obrigatório) O protocolo usado para se comunicar com os agentes. Os valores possíveis são `"SSL"` ou `"PLAINTEXT"`.
- `"topicName"` (Obrigatório): uma lista separada por vírgulas de tópicos para assinar. É necessário especificar um e apenas um de `"topicName"`, `"assign"` ou `"subscribePattern"`.
- `"assign"` (Obrigatório): uma string JSON para especificar o `TopicPartitions` a ser consumido. É necessário especificar um e apenas um de `"topicName"`, `"assign"` ou `"subscribePattern"`.

  Exemplo: '{"topicA":[0,1],"topicB":[2,4]}'

- "subscribePattern": (obrigatório) uma string regex Java que identifica a lista de tópicos para assinar. É necessário especificar um e apenas um de "topicName", "assign" ou "subscribePattern".

  Exemplo: 'topic.*'

- "classification" (Obrigatório) O formato de arquivo usado pelos dados no registro. Obrigatório, a menos que fornecido por meio do catálogo de dados.
- "delimiter" (Opcional) O separador de valores usado quando a classification é CSV. O padrão é ",".
- "startingOffsets": (opcional) a posição inicial no tópico do Kafka de onde ler os dados. Os valores possíveis são "earliest" ou "latest". O valor padrão é "latest".
- "startingTimestamp": (Opcional, compatível somente com o AWS Glue versão 4.0 ou posterior) O timestamp do registro no tópico do Kafka do qual ler os dados. O valor possível é uma string de timestamp no formato UTC no padrão yyyy-mm-ddTHH:MM:SSZ (onde Z representa um desvio do fuso horário UTC com +/-). Por exemplo: "2023-04-04T08:00:00-04:00").

  Observação: somente um dos 'startingOffsets' ou 'startingTimestamp' pode estar presente na lista de opções de conexão do script de streaming AWS Glue. Incluir essas duas propriedades resultará em falha no trabalho.

- "endingOffsets": (opcional) o ponto final quando uma consulta em lote é encerrada. Os valores possíveis são "latest" ou uma string JSON que especifica um deslocamento final para cada TopicPartition.

  Para a string JSON, o formato é {"topicA":{"0":23,"1":-1},"topicB":{"0":-1}}. O valor -1 como um deslocamento representa "latest".

- "pollTimeoutMs": (opcional) o tempo limite em milissegundos para sondar dados do Kafka em executores de trabalho do Spark. O valor padrão é 512.
- "numRetries": (opcional) o número de novas tentativas antes de falhar em obter os deslocamentos do Kafka. O valor padrão é 3.
- "retryIntervalMs": (opcional) o tempo em milissegundos a se esperar antes de tentar novamente buscar os deslocamentos do Kafka. O valor padrão é 10.
- "maxOffsetsPerTrigger": (opcional) o limite de taxa no número máximo de deslocamentos que são processados por intervalo do acionador. O número total especificado de deslocamentos é dividido proporcionalmente entre topicPartitions de diferentes volumes. O valor padrão

é nulo, o que significa que o consumidor lê todos os deslocamentos até o deslocamento mais recente conhecido.

- `"minPartitions"`: (opcional) o número mínimo desejado de partições a serem lidas do Kafka. O valor padrão é nulo, o que significa que o número de partições do Spark é igual ao número de partições do Kafka.
- `"includeHeaders"`: (opcional) informa se deve incluir os cabeçalhos do Kafka. Quando a opção estiver definida como "true", a saída de dados conterá uma coluna adicional chamada "glue_streaming_kafka_headers" com o tipo `Array[Struct(key: String, value: String)]`. O valor padrão é “false”. Essa opção está disponível no AWS Glue versão 3.0 ou posterior.
- `"schema"` (Obrigatório quando inferSchema é definido como false): o esquema a ser usado para processar a carga útil. Se a classificação for `avro`, o esquema fornecido deverá estar no formato de esquema Avro. Se a classificação não for `avro`, o esquema fornecido deverá estar no formato de esquema DDL.

  Veja a seguir alguns exemplos de esquema.

  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): o valor padrão é "false". Se definido como “true”, o esquema será detectado em runtime com base na carga útil em `foreachbatch`.
- `"avroSchema"` (Descontinuado): parâmetro usado para especificar um esquema de dados Avro quando o formato Avro é usado. Esse parâmetro foi descontinuado. Use o parâmetro `schema`.
- `"addRecordTimestamp"`: (opcional) quando essa opção for definida como “true“, a saída de dados conterá uma coluna adicional denominada “__src_timestamp” que indica a hora em que o registro correspondente foi recebido pelo tópico. O valor padrão é "false". Essa opção é compatível com o AWS Glue versão 4.0 ou posterior.
- `"emitConsumerLagMetrics"`: (Opcional) Quando a opção for definida como 'verdadeira', para cada lote, ela emitirá as métricas da duração entre o registro mais antigo recebido pelo tópico e o horário em AWS Glue que ele chegou. CloudWatch O nome da métrica é “glue.driver.streaming”. maxConsumerLagInMs”. O valor padrão é "false". Essa opção é compatível com o AWS Glue versão 4.0 ou posterior.

Ao gravar, use as seguintes opções de conexão com o `"connectionType": "kafka"`:

- `"connectionName"`(Obrigatório) Nome da conexão AWS Glue usada para se conectar ao cluster Kafka (semelhante à fonte do Kafka).
- `"topic"` (Obrigatório) Se uma coluna de tópico existir, seu valor será usado como tópico ao gravar a linha especificada no Kafka, a menos que a opção de configuração do tópico esteja definida. Ou seja, a opção de configuração `topic` substitui a coluna do tópico.
- `"partition"` (Opcional) Se um número de partição válido for especificado, essa `partition` será usada no envio do registro.

  Se nenhuma partição for especificada, mas uma `key` estiver presente, uma partição será escolhida usando um hash da chave.

  Se nem `key` nem `partition` estiverem presentes, uma partição será escolhida com base no particionamento fixo dessas alterações quando pelo menos bytes batch.size forem produzidos na partição.

- `"key"` (Opcional) Usado para particionar se `partition` for nulo.
- `"classification"` (Opcional) O formato de arquivo usado pelos dados no registro. Só oferecemos suporte a JSON, CSV e Avro.

  Com o formato Avro, podemos fornecer um avroSchema personalizado para serializar, mas observe que isso também precisa ser fornecido na fonte para desserialização. Caso contrário, por padrão, ele usa o Apache AvroSchema para serializar.

Além disso, você pode ajustar o coletor Kafka conforme necessário atualizando os parâmetros de configuração do produtor Kafka. Observe que não há nenhuma lista de permissões nas opções de conexão. Todos os pares de chave-valor são mantidos no coletor como estão.

No entanto, há uma pequena lista de opções negadas que não entrarão em vigor. Para obter mais informações, consulte Configurações específicas do Kafka.

Conexões do Kinesis

Você pode ler e gravar o Amazon Kinesis Data Streams usando informações armazenadas em uma tabela do Data Catalog ou fornecendo informações para acessar diretamente o fluxo de dados. Você pode ler informações do Kinesis em um Spark DataFrame e depois convertê-las em um Glue. AWS DynamicFrame Você pode DynamicFrames gravar no Kinesis em um formato JSON. Se você acessar diretamente o fluxo de dados, use essas opções para fornecer as informações sobre como acessar o fluxo de dados.

Se você usar `getCatalogSource` ou `create_data_frame_from_catalog` para consumir registros de uma fonte de transmissão do Kinesis, o trabalho tem o banco de dados do catálogo de dados e as informações de nome da tabela, e pode usá-los para obter alguns parâmetros básicos para leitura da fonte de transmissão do Kinesis. Se você usar `getSource`, `getSourceWithFormat`, `createDataFrameFromOptions` ou `create_data_frame_from_options`, será necessário especificar esses parâmetros básicos usando as opções de conexão descritas aqui.

Você pode especificar as opções de conexão para o Kinesis usando os seguintes argumentos para os métodos especificados na classe `GlueContext`.

- Scala
  - `connectionOptions`: usar com `getSource`, `createDataFrameFromOptions`, `getSink`
  - `additionalOptions`: usar com `getCatalogSource`, `getCatalogSink`

  - `options`: usar com `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: usar com `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: usar com `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: usar com `getSource`, `getSink`

Para notas e restrições sobre trabalhos de ETL de streaming, consulte the section called “Notas e restrições sobre ETL de transmissão”.

Configurar o Kinesis

Para se conectar a um stream de dados do Kinesis em uma tarefa do AWS Glue Spark, você precisará de alguns pré-requisitos:

- Se estiver lendo, o trabalho AWS Glue deve ter permissões IAM de nível de acesso de leitura ao stream de dados do Kinesis.
- Se estiver gravando, o trabalho do AWS Glue deve ter permissões IAM no nível de acesso Write ao stream de dados do Kinesis.

Em certos casos, você precisará configurar pré-requisitos adicionais:

- Se sua tarefa do AWS Glue estiver configurada com conexões de rede adicionais (normalmente para se conectar a outros conjuntos de dados) e uma dessas conexões fornecer opções de rede Amazon VPC, isso direcionará seu trabalho para se comunicar pela Amazon VPC. Nesse caso, você também precisará configurar o fluxo de dados do Kinesis para se comunicar pela Amazon VPC. É possível fazer isso criando um endpoint da VPC de interface entre a Amazon VPC e o fluxo de dados do Kinesis. Para obter mais informações, consulte Using Kinesis Data Streams with Interface VPC Endpoints.
- Ao especificar Amazon Kinesis Data Streams em outra conta, você deve configurar os perfis e políticas para permitir o acesso entre contas. Para obter mais informações, consulte Exemplo: Ler de uma transmissão do Kinesis em outra conta.

Para obter mais informações sobre pré-requisitos de trabalho de ETL de streaming, consulte the section called “Trabalhos de transmissão de ETL”.

Ler no Kinesis

Exemplo: ler de fluxos do Kinesis

Usado em conjunto com the section called “forEachBatch”.

Exemplo para fonte de transmissão do 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)
```

Gravar no Kinesis

Exemplo: gravação em streams do Kinesis

Usado em conjunto com the section called “forEachBatch”. Você DynamicFrame será gravado no stream em formato JSON. Se o trabalho não puder ser gravado após várias tentativas, ele falhará. Por padrão, cada DynamicFrame registro será enviado para o stream do Kinesis individualmente. Você pode configurar esse comportamento usando `aggregationEnabled` parâmetros associados.

Exemplo de gravação para o Amazon Kinesis a partir de um trabalho de streaming:

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)
```

Parâmetros de conexão do Kinesis

Designa opções de conexão para o Amazon Kinesis Data Streams.

Use as seguintes opções de conexão para fontes de dados de transmissão do Kinesis:

- `"streamARN"` (obrigatório) usado para leitura/gravação. O ARN do fluxo de dados do Kinesis.
- `"classification"` (Obrigatório para leitura) Usado para leitura. O formato de arquivo usado pelos dados no registro. Obrigatório, a menos que fornecido por meio do catálogo de dados.
- `"streamName"` (opcional) usado para leitura. O nome de um fluxo de dados do Kinesis de onde ler. Usado com `endpointUrl`.
- `"endpointUrl"` (opcional) usado para leitura. Padrão: “https://kinesis.us-east-1.amazonaws.com” O AWS endpoint do stream do Kinesis. Você não precisa alterar isso, a menos que esteja se conectando a uma região especial.
- `"partitionKey"` (opcional) usado para gravação. A chave de partição do Kinesis usada na produção de registros.
- `"delimiter"` (opcional) usado para leitura. O separador de valores usado quando a `classification` é CSV. O padrão é `","`.
- `"startingPosition"`: (opcional) usado para leitura. A posição inicial no fluxo de dados do Kinesis de onde ler os dados. Os valores possíveis são `"latest"`, `"trim_horizon"`, `"earliest"` ou uma string de timestamp no formato UTC no padrão `yyyy-mm-ddTHH:MM:SSZ` (onde `Z` representa um desvio do fuso horário UTC com +/-). Por exemplo: "2023-04-04T08:00:00-04:00"). O valor padrão é `"latest"`. Observação: a string Timestamp no formato UTC para `"startingPosition"` é compatível somente com a versão 4.0 ou posterior do AWS Glue.

- `"failOnDataLoss"`: (Opcional) Falha na tarefa se algum fragmento ativo estiver ausente ou expirado. O valor padrão é `"false"`.
- `"awsSTSRoleARN"`: (opcional) usado para leitura/gravação. O Amazon Resource Name (ARN) da função a ser assumida usando AWS Security Token Service ()AWS STS. Essa função deve ter permissões para descrever ou ler operações de registro para o fluxo de dados do Kinesis. Você deve usar esse parâmetro ao acessar um fluxo de dados em uma conta diferente. Usado em conjunto com `"awsSTSSessionName"`.
- `"awsSTSSessionName"`: (opcional) usado para leitura/gravação. Um identificador para a sessão que assume a função usando o AWS STS. Você deve usar esse parâmetro ao acessar um fluxo de dados em uma conta diferente. Usado em conjunto com `"awsSTSRoleARN"`.
- `"awsSTSEndpoint"`: (Opcional) O AWS STS endpoint a ser usado ao se conectar ao Kinesis com uma função assumida. Isso permite usar o AWS STS endpoint regional em uma VPC, o que não é possível com o endpoint global padrão.
- `"maxFetchTimeInMs"`: (opcional) usado para leitura. O tempo máximo para o executor do trabalho ler registros referentes ao lote atual do fluxo de dados do Kinesis especificado em milissegundos (ms). Várias chamadas de API `GetRecords` podem ser feitas nesse período. O valor padrão é `1000`.
- `"maxFetchRecordsPerShard"`: (opcional) usado para leitura. O número máximo de registros a serem obtidos por fragmento no fluxo de dados do Kinesis por microlote. Observação: o cliente poderá exceder esse limite se o trabalho de streaming já tiver lido registros extras do Kinesis (na mesma chamada get-records). Se `maxFetchRecordsPerShard` precisa ser rigoroso, então precisa ser um múltiplo de `maxRecordPerRead`. O valor padrão é `100000`.
- `"maxRecordPerRead"`: (opcional) usado para leitura. O número máximo de registros a serem obtidos por fragmento no fluxo de dados do Kinesis em cada operação `getRecords`. O valor padrão é `10000`.
- `"addIdleTimeBetweenReads"`: (opcional) usado para leitura. Adiciona um atraso de tempo entre duas operações `getRecords`. O valor padrão é `"False"`. Essa opção só pode ser configurada para o Glue versão 2.0 e posterior.
- `"idleTimeBetweenReadsInMs"`: (opcional) usado para leitura. O atraso mínimo entre duas operações , especificado em ms. O valor padrão é `1000`. Essa opção só pode ser configurada para o Glue versão 2.0 e posterior.
- `"describeShardInterval"`: (opcional) usado para leitura. O intervalo de tempo mínimo entre duas chamadas de API `ListShards` para que seu script considere a refragmentação. Para

obter mais informações, consulte Estratégias para refragmentação no Guia do desenvolvedor do Amazon Kinesis Data Streams. O valor padrão é `1s`.

- `"numRetries"`: (opcional) usado para leitura. O número máximo de novas tentativas para solicitações de API do Kinesis Data Streams. O valor padrão é `3`.
- `"retryIntervalMs"`: (opcional) usado para leitura. O período de espera (especificado em ms) antes de repetir a chamada da API Kinesis Data Streams. O valor padrão é `1000`.
- `"maxRetryIntervalMs"`: (opcional) usado para leitura. O período de espera máximo (especificado em ms) entre duas tentativas de uma chamada de API Kinesis Data Streams. O valor padrão é `10000`.
- `"avoidEmptyBatches"`: (opcional) usado para leitura. Evita a criação de um trabalho de micro lote vazio verificando se há dados não lidos no fluxo de dados do Kinesis antes de o lote ser iniciado. O valor padrão é `"False"`.
- `"schema"`: (Obrigatório quando inferSchema é definido como false): usado para leitura. O esquema a ser usado para processar a carga. Se a classificação for `avro`, o esquema fornecido deverá estar no formato de esquema Avro. Se a classificação não for `avro`, o esquema fornecido deverá estar no formato de esquema DDL.

  Veja a seguir alguns exemplos de esquema.

  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) usado para leitura. O valor padrão é "false". Se definido como “true”, o esquema será detectado em runtime com base na carga útil em `foreachbatch`.
- `"avroSchema"`: (Obsoleto) Usado para leitura. Parâmetro usado para especificar um esquema de dados Avro quando o formato Avro é usado. Esse parâmetro foi descontinuado. Use o parâmetro `schema`.
- `"addRecordTimestamp"`: (opcional) usado para leitura. Quando essa opção for definida como "true", a saída de dados conterá uma coluna adicional denominada "__src_timestamp" que indica a hora que o registro correspondente é recebido pelo fluxo. O valor padrão é "false". Essa opção é compatível com o AWS Glue versão 4.0 ou posterior.
- `"emitConsumerLagMetrics"`: (opcional) usado para leitura. Quando a opção é definida como “verdadeira”, para cada lote, ela emitirá as métricas da duração entre o registro mais antigo recebido pelo stream e o horário em AWS Glue que ele chega. CloudWatch O nome da métrica é “glue.driver.streaming”. maxConsumerLagInMs”. O valor padrão é "false". Essa opção é compatível com o AWS Glue versão 4.0 ou posterior.
- `"fanoutConsumerARN"`: (opcional) usado para leitura. O ARN de um consumidor de fluxo do Kinesis para o fluxo especificado em `streamARN`. Usado para habilitar o modo de distribuição avançada para a conexão do Kinesis. Para obter mais informações sobre como consumir um fluxo do Kinesis com distribuição avançada, consulte the section called “Usar distribuição avançada nas tarefas de streaming do Kinesis”.
- `"recordMaxBufferedTime"` (opcional) usado para gravação. Padrão: 1000 (ms). Tempo máximo em que um registro é armazenado em buffer enquanto espera para ser gravado.
- `"aggregationEnabled"` (opcional) usado para gravação. Padrão: true. Especifica se os registros devem ser agregados antes de serem enviados para o Kinesis.
- `"aggregationMaxSize"` (opcional) usado para gravação. Padrão: 51200 (bytes) Se um registro for maior que esse limite, ele ignorará o agregador. Nota: O Kinesis impõe um limite de 50 KB no

tamanho do registro. Se você definir isso além de 50 KB, registros grandes serão rejeitados pelo Kinesis.

- `"aggregationMaxCount"` (opcional) usado para gravação. Padrão: 4294967295. O número máximo de itens a serem retornados em um registro agregado.
- `"producerRateLimit"` (opcional) usado para gravação. Padrão: 150 (%). Limita o throughput por fragmento enviado por um único produtor (como seu trabalho), como uma porcentagem do limite de back-end.
- `"collectionMaxCount"` (opcional) usado para gravação. Padrão: 500. Número máximo de itens a serem embalados em uma PutRecords solicitação.
- `"collectionMaxSize"` (opcional) usado para gravação. Padrão: 5242880 (bytes) Quantidade máxima de dados a serem enviados com uma PutRecords solicitação.

AWS Glue Opções de streaming

Designa uma conexão com um cluster do Kafka ou um cluster do Amazon Managed Streaming for Apache Kafka.

É possível ler e gravar fluxos de dados do Kafka usando informações armazenadas em uma tabela do Catálogo de Dados ou fornecendo informações para acessar diretamente o fluxo de dados. Você pode ler informações de Kafka em um Spark e depois DataFrame convertê-las em um Glue. AWS DynamicFrame Você pode DynamicFrames escrever no Kafka no formato JSON. Se você acessar diretamente o fluxo de dados, use essas opções para fornecer as informações sobre como acessar o fluxo de dados.

Se você usar `getCatalogSource` ou `create_data_frame_from_catalog` para consumir registros de uma fonte de streaming do Kafka, ou `getCatalogSink` ou `write_dynamic_frame_from_catalog` para gravar registros no Kafka, e o trabalho tiver o banco de dados do catálogo de dados e as informações de nome da tabela, e poderá usá-los para obter alguns parâmetros básicos para leitura da fonte de streaming do Kafka. Se você usar `getSource`, `getCatalogSink`, `getSourceWithFormat`, `getSinkWithFormat`, `createDataFrameFromOptions`, `create_data_frame_from_options` ou `write_dynamic_frame_from_catalog`, será necessário especificar esses parâmetros básicos usando as opções de conexão descritas aqui.

Você pode especificar as opções de conexão para o Kafka usando os seguintes argumentos para os métodos especificados na classe `GlueContext`.

- Scala
  - `connectionOptions`: usar com `getSource`, `createDataFrameFromOptions`, `getSink`
  - `additionalOptions`: usar com `getCatalogSource`, `getCatalogSink`
  - `options`: usar com `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: usar com `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: usar com `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: usar com `getSource`, `getSink`

Para notas e restrições sobre trabalhos de ETL de streaming, consulte the section called “Notas e restrições sobre ETL de transmissão”.

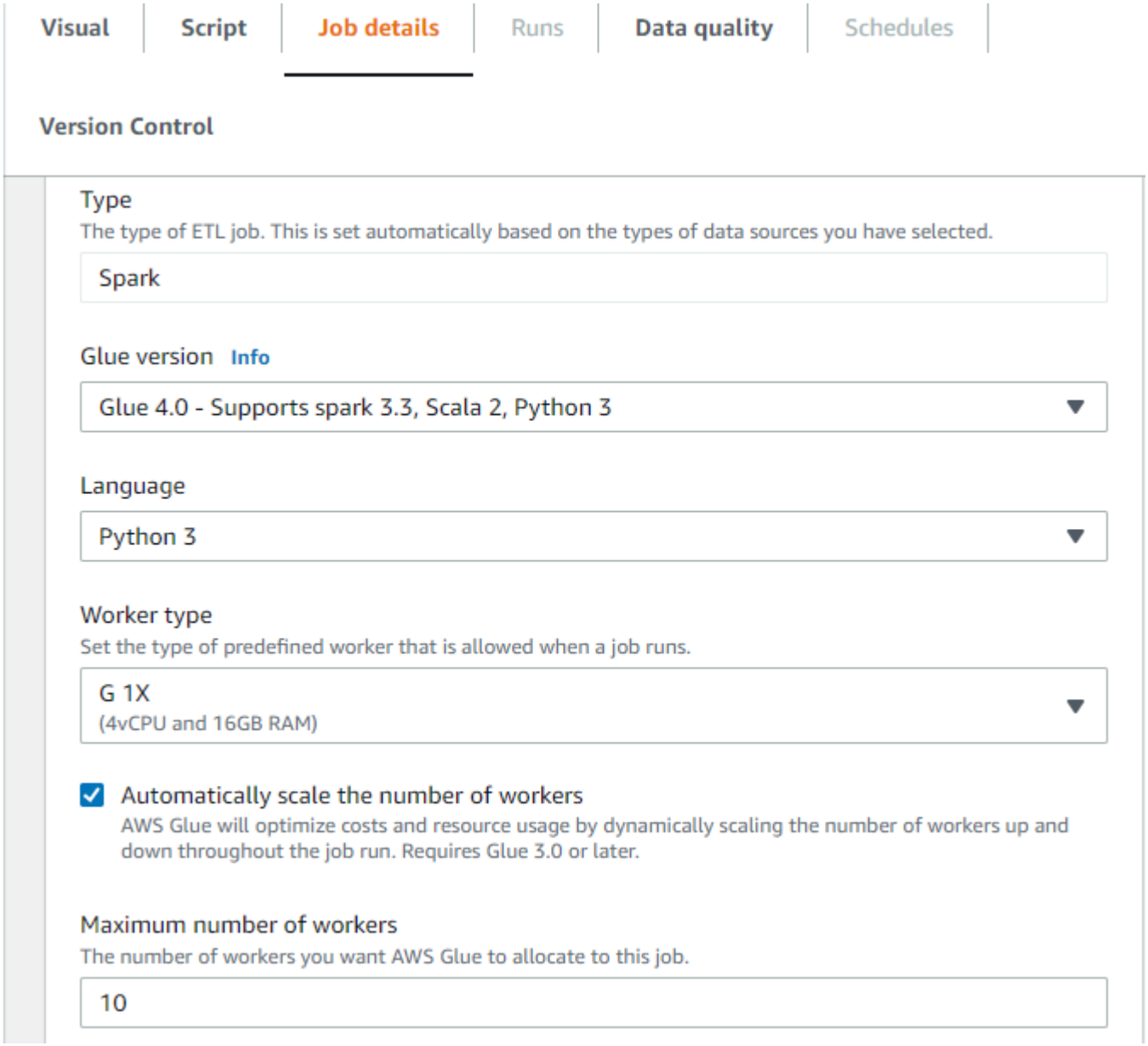
Ajuste de escala automático de streaming do AWS Glue

As seções apresentadas a seguir fornecem informações sobre o ajuste de escala automático de streaming do AWS Glue.

Habilitar Auto Scaling no AWS Glue Studio

Na guia Job details (Detalhes do trabalho) no AWS Glue Studio, escolha o tipo Spark ou Spark Streaming (Streaming do Spark) e a Glue version (Versão do Glue) como **`Glue 3.0`** ou **`Glue 4.0`**. Uma caixa de seleção será exibida abaixo de Worker type (Tipo de operador).

- Selecione a opção Automatically scale the number of workers (Dimensionar automaticamente o número de operadores).
- Defina Maximum numer of workers (Número máximo de operadores) para estabelecer o número máximo de operadores que podem ser transferidos para a execução do trabalho.



Habilitar o Auto Scaling com a AWS CLI ou SDK

Para habilitar o Auto Scaling diretamente da AWS CLI para sua execução de trabalho, execute `start-job-run` com a seguinte configuração:

```
{
    "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...
}
```

Quando a execução do trabalho de ETL estiver concluída, também será possível chamar `get-job-run` para verificar o uso efetivo de recurso da execução de trabalho em segundos de DPU. Observação: o novo campo DPUSeconds só aparecerá para trabalhos em lote no AWS Glue 3.0 ou posterior habilitado com o Auto Scaling. Esse campo não é compatível com trabalhos de transmissão.

```
$ 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
    }
}
```

Você também pode configurar execuções de tarefas com o Auto Scaling usando o AWS Glue SDK com a mesma configuração.

Como funciona

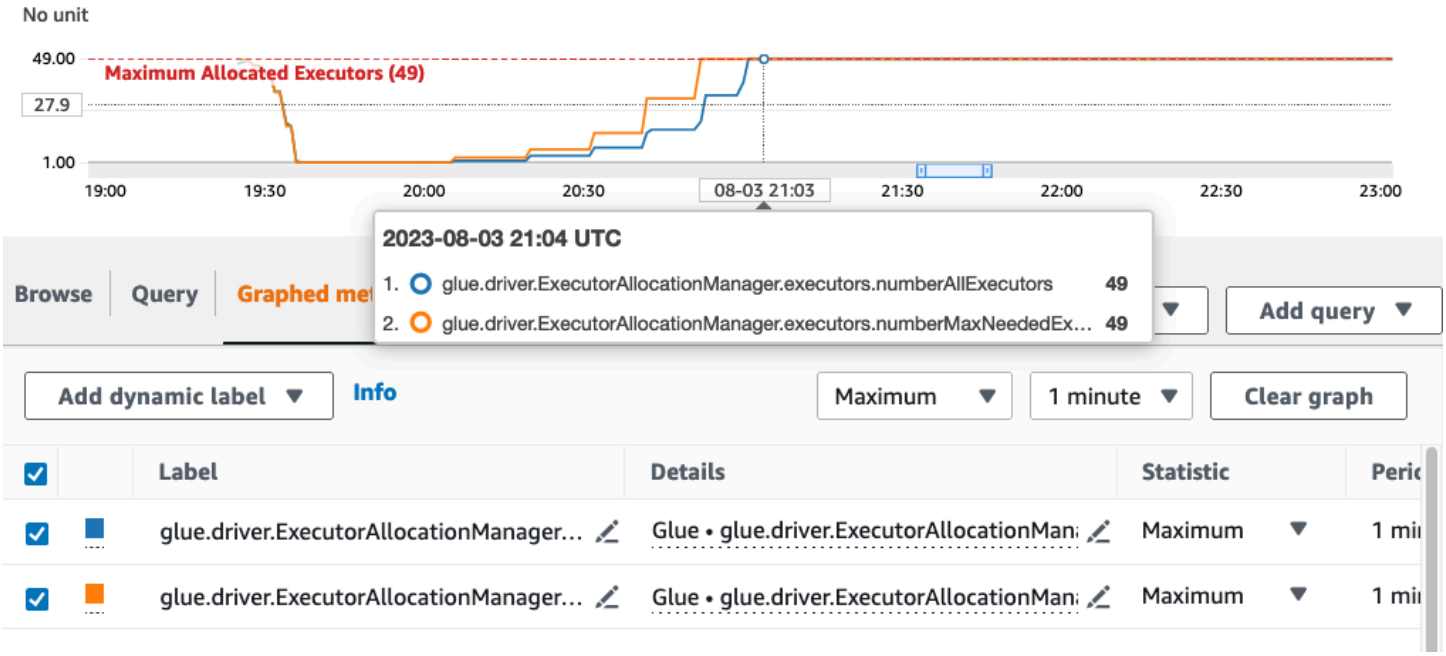
Escalar entre micro lotes

O exemplo apresentado a seguir é usado para descrever como funciona o ajuste de escala automático.

- Você tem um trabalho do AWS Glue que começa com 50 DPUs.
- O ajuste de escala automático está habilitado.

Neste exemplo, o AWS Glue analisa a métrica “batchProcessingTimeInMs” para alguns micro lotes e determina se os trabalhos estão sendo concluídos dentro do tamanho da janela que você estabeleceu. Se os trabalhos forem concluídos com antecedência e, com base em quanto tempo forem concluídos, o AWS Glue poderá reduzir a escala verticalmente. Essa métrica, plotada com “numberAllExecutors”, pode ser monitorada no Amazon CloudWatch para visualizar como funciona o ajuste de escala automático.

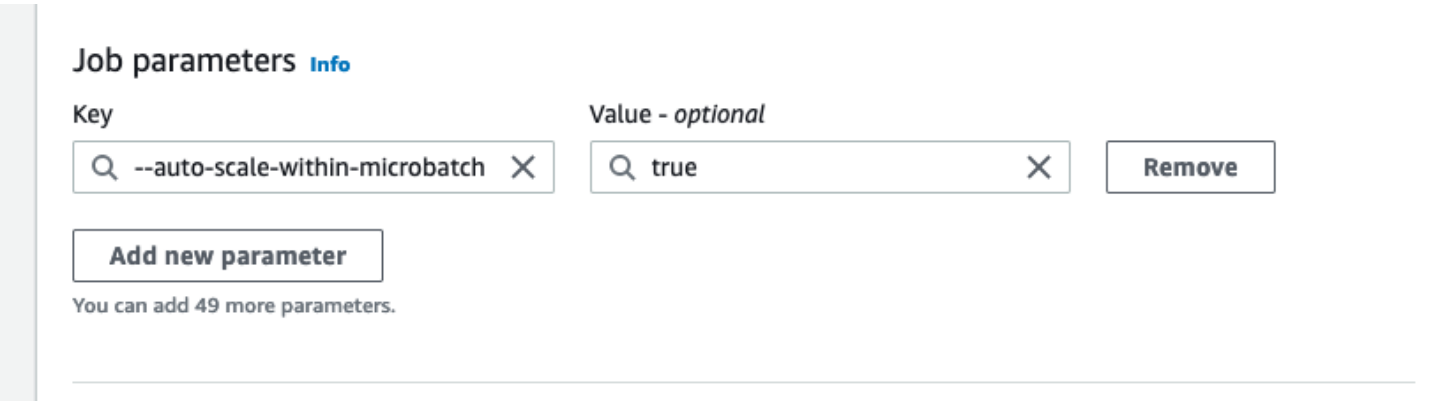
O número de executores aumenta ou reduz a escala verticalmente de forma exponencial somente após a conclusão de cada micro lote. Como você pode visualizar no log de monitoramento do Amazon CloudWatch, o AWS Glue analisa o número de executores necessários (linha laranja) e escala os executores (linha azul) para corresponder a eles automaticamente.



Depois que o AWS Glue reduzir a escala verticalmente para o número de executores e observar que os volumes de dados aumentam, consequentemente aumentando o tempo de processamento do micro lote, o AWS Glue aumentará a escala verticalmente para 50 DPUs, que é o limite superior especificado.

Escalar em micro lotes

No exemplo acima, o sistema monitora alguns micro lotes concluídos para tomar uma decisão sobre reduzir ou aumentar a escala verticalmente. Janelas mais longas requerem um ajuste de escala automático para responder mais rapidamente dentro do micro lote, em vez de esperar por alguns micro lotes. Para estes casos, você pode usar uma configuração adicional `--auto-scaIe-within-microbatch` como `true`. É possível adicionar essa configuração às propriedades do trabalho do AWS Glue no AWS Glue Studio, conforme mostrado abaixo.



Janelas de manutenção para AWS Glue streaming

AWS Glue realiza periodicamente atividades de manutenção. Durante essas janelas de manutenção, AWS Glue será necessário reiniciar seus trabalhos de streaming. Você pode controlar quando os trabalhos são reiniciados especificando janelas de manutenção. Nesta seção, descrevemos onde você pode configurar a janela de manutenção e os comportamentos específicos que você deve considerar.

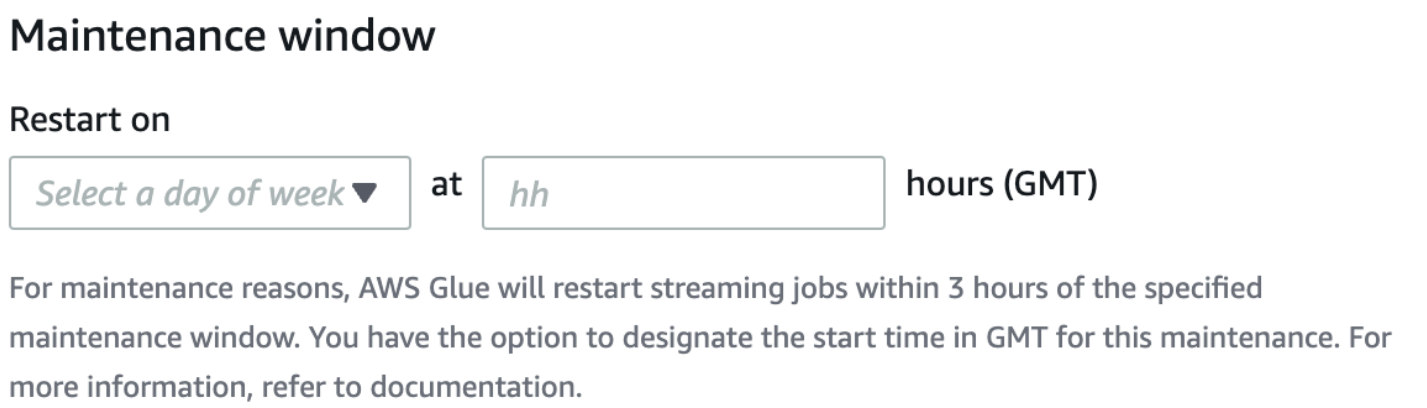
Tópicos

Configurando uma janela de manutenção

Você pode configurar uma janela de manutenção usando o AWS Glue Studio ou as APIs.

Configurando uma janela de manutenção no AWS Glue Studio

Você pode especificar uma janela de manutenção na página Detalhes do Job do seu job de AWS Glue Streaming. Você pode especificar o dia e a hora em GMT. AWS Glue reiniciará seu trabalho dentro da janela de tempo especificada.



Configurando uma janela de manutenção na API

Como alternativa, você pode configurar a janela de manutenção na API Create Job. Aqui está um exemplo de configuração de janelas de manutenção por meio da API.

```
aws glue create-job –name jobName –role roleArnForTheJob –command
 Name=gluestreaming,ScriptLocation=s3-path-to-the-script --maintenance-window="Sun:10"
```

Um exemplo de comando é o seguinte:

```
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
```

Comportamento da janela de manutenção

AWS Glue passa por uma série de etapas para decidir quando reiniciar um trabalho:

1. Quando um novo trabalho de streaming é iniciado, AWS Glue primeiro verifica se há um tempo limite associado à execução do trabalho. Um tempo limite permite que você configure a hora de término do trabalho. Se o tempo limite for inferior a 7 dias, o trabalho não será reiniciado.
2. Se o tempo limite for maior que 7 dias, AWS Glue verifique se a janela de manutenção está configurada para o trabalho. Se estiver, essa janela será selecionada e a janela será atribuída à execução do trabalho. AWS Glue reiniciará o trabalho dentro de 3 horas da janela de manutenção especificada. Por exemplo, se você configurar a janela de manutenção para segunda-feira às 10h GMT, seus trabalhos serão reiniciados entre 10h GMT e 13h GMT.

3. Se a janela de manutenção não estiver configurada, AWS Glue definirá automaticamente o horário de reinicialização para 7 dias após o horário de início da execução do trabalho. Por exemplo, se você iniciou seu trabalho em 01/07/2024 às 12:00 GMT e não especificou janelas de manutenção, seu trabalho será configurado para reiniciar em 7/8/2024 às 12:00 GMT.

> **Note**
>
> Se você já estiver executando trabalhos de streaming, essa alteração afetará você a partir de 1º de julho de 2024. Você terá tempo até 30 de junho para configurar suas janelas de manutenção. Depois de 1º de julho, todos os trabalhos de streaming que você iniciar serão reiniciados de acordo com esta documentação. Se precisar de suporte adicional, entre em contato com o AWS Support.

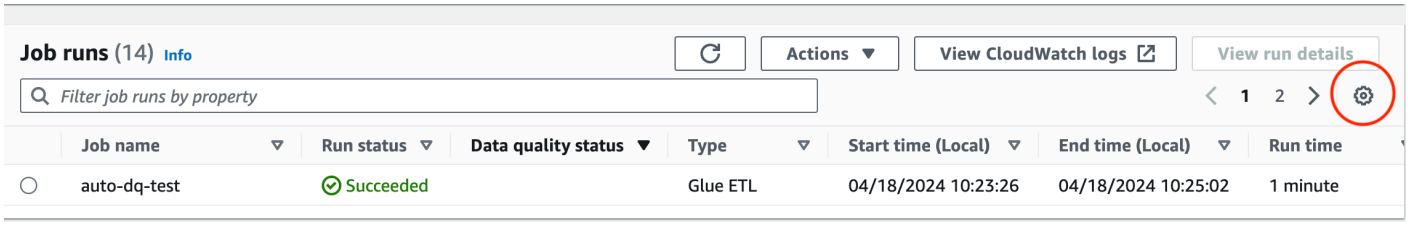
4. Às vezes, AWS Glue talvez não seja possível reiniciar o trabalho, especialmente quando o microlote em andamento não é processado. Nesses casos, o trabalho não será interrompido. Nesses casos, AWS Glue reiniciará o trabalho após 14 dias e, nesse caso, a janela de manutenção não será respeitada.

Monitoramento de trabalhos

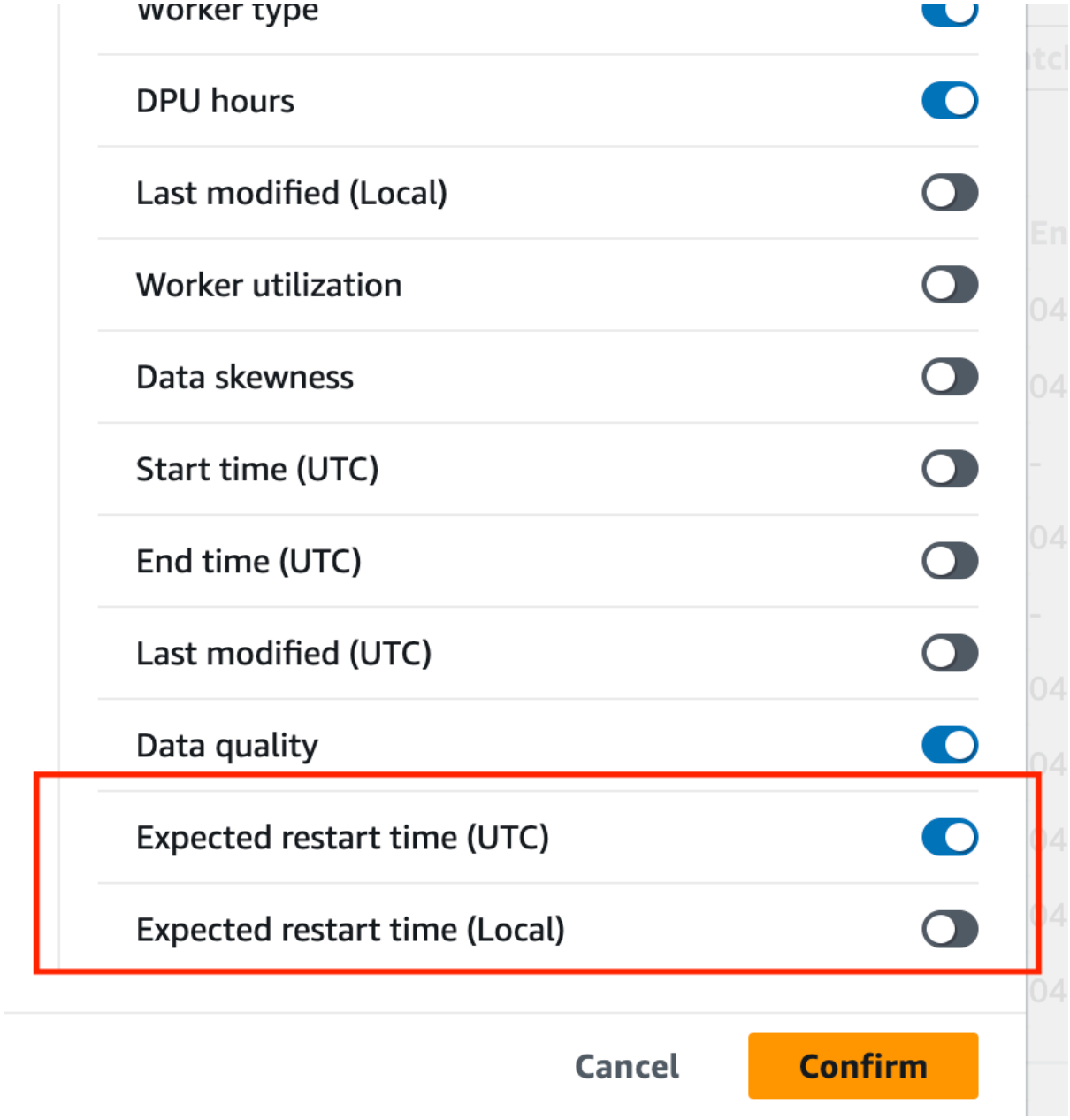
Você pode monitorar os trabalhos na página AWS Glue Studio Monitoring.

Para ver o horário esperado para a próxima reinicialização dos trabalhos de streaming, mostre a coluna na tabela Job runs na página Monitoramento.

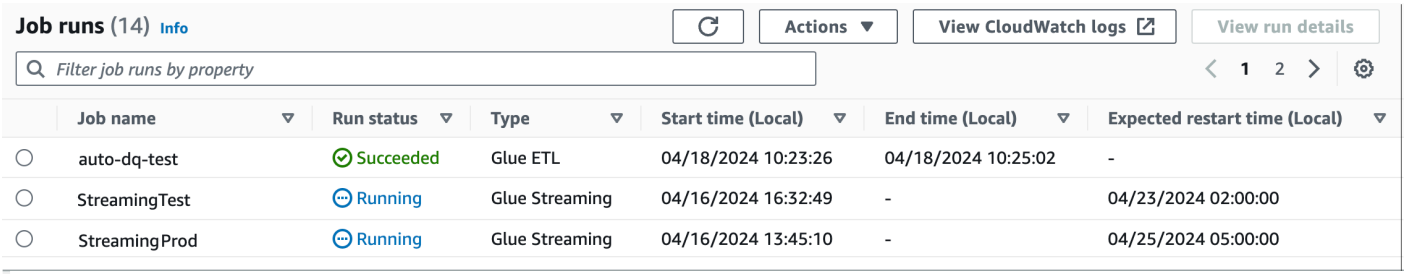
1. Clique no ícone de engrenagem no canto superior direito da tabela.



2. Role para baixo e ative a coluna Tempo de reinicialização esperado. As opções de horário UTC e local estão disponíveis.



3. Em seguida, você pode visualizar as colunas na tabela.



O trabalho original terá o status “EXPIRADO” e a nova instância do trabalho terá o status “EM EXECUÇÃO”. A nova execução da tarefa que foi reiniciada terá uma ID de execução da tarefa como uma concatenação da ID de execução inicial da tarefa mais o prefixo “restart_” representando a contagem de reinicializações. Por exemplo, se a ID de execução inicial da tarefa for`jr_1234`, a execução da tarefa reiniciada terá a ID `jr1234_restart_1` da primeira reinicialização. A segunda reinicialização será `jr1234_restart_2` para a segunda reinicialização e assim por diante.

Sua tentativa de nova tentativa não será afetada por causa das reinicializações. Se uma execução falhar e uma nova execução for iniciada devido a uma nova tentativa automática, o contador de reinicialização iniciará novamente em 1. Por exemplo, se uma execução falhar em`jr_1234_attempt_3_restart_5`, uma nova tentativa automática iniciará uma nova execução com ID: `jr_id1_attempt_4` e quando essa tentativa for reiniciada após 7 dias, a nova ID de execução será. `jr_id1_attempt_4_restart_1`

Tratamento da perda de dados

Durante a reinicialização da manutenção, o AWS Glue Streaming segue um processo que garante a integridade e a consistência dos dados entre a execução da tarefa anterior e a execução da tarefa reiniciada. Observe que isso AWS Glue não garante a integridade e a consistência dos dados entre as reinicializações das tarefas. Recomendamos considerar a arquitetura para lidar com dados duplicados nas tarefas de streaming.

1. Detectando condições de reinicialização da manutenção: o AWS Glue streaming monitora as condições que indicam quando uma reinicialização da manutenção deve ser acionada, como quando uma janela de manutenção é atingida após 7 dias ou uma reinicialização forçada é necessária após 14 dias.
2. Invocando uma rescisão normal: quando as condições de reinicialização da manutenção são atendidas, o AWS Glue Streaming inicia um processo de encerramento normal para o trabalho em execução no momento. Esse processo envolve as seguintes etapas:
   a. Interromper a ingestão de novos dados: o trabalho de streaming para de consumir novos dados das fontes de entrada (por exemplo, tópicos do Kafka, streams do Kinesis ou arquivos).
   b. Processamento de dados pendentes: o trabalho continua processando todos os dados que já estão presentes em seus buffers ou filas internas.
   c. Confirmação de compensações e pontos de verificação: o trabalho transfere as compensações ou pontos de verificação mais recentes para sistemas externos (por exemplo, Kafka, Kinesis ou Amazon S3) para garantir que o trabalho reiniciado possa continuar de onde o trabalho anterior parou.

3. Reiniciando o trabalho: após a conclusão do processo de encerramento normal, o AWS Glue Streaming reinicia o trabalho usando o estado preservado e os pontos de verificação. A tarefa reiniciada retoma o processamento do último offset ou ponto de verificação confirmado, garantindo que nenhum dado seja perdido ou duplicado.
4. Retomando o processamento de dados: o trabalho reiniciado retoma o processamento de dados a partir do ponto em que o trabalho anterior parou. Ele continua ingerindo novos dados das fontes de entrada, a partir do último offset ou ponto de verificação confirmado, e processa os dados de acordo com a lógica de ETL definida.

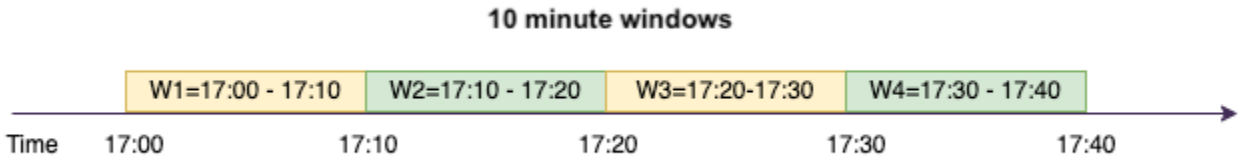
Conceitos avançados de streaming do AWS Glue

Em aplicações contemporâneas orientadas a dados, a importância dos dados diminui ao longo do tempo e o seu valor passa de preditivo a reativo. Como resultado, os clientes desejam processar dados em tempo real para tomar decisões mais rápidas. Ao lidar com feeds de dados em tempo real, como os provenientes de sensores de IoT, os dados podem ser apresentados de forma desordenada ou sofrer atrasos no processamento devido à latência da rede e outras falhas relacionadas à fonte durante a ingestão. Como parte da plataforma do AWS Glue, o AWS Glue Streaming aproveita essas funcionalidades para fornecer ETL de streaming escalável e com tecnologia sem servidor, com base no streaming estruturado do Apache Spark, capacitando os usuários com processamento de dados em tempo real.

Neste tópico, exploraremos as funcionalidades e os conceitos avançados de streaming do AWS Glue Streaming.

Considerações sobre o tempo ao processar fluxos

Existem quatro noções de tempo no momento do processamento de fluxos:



- Horário do evento: o horário em que o evento ocorreu. Na maioria dos casos, este campo está incorporado aos próprios dados do evento, na fonte.

- E vent-time-window — O intervalo de tempo entre dois horários de eventos. Conforme mostrado no diagrama acima, W1 é das event-time-window 17:00 às 17:10. Cada um event-time-window é um agrupamento de vários eventos.
- Horário do acionamento: o horário do acionamento controla a frequência com que ocorre o processamento dos dados e a atualização dos resultados. Este é o horário em que o processamento do micro lote foi iniciado.
- Horário da ingestão: o horário em que os dados de fluxo foram ingeridos no serviço de streaming. Se o horário do evento não estiver incorporado no próprio evento, esse horário poderá ser usado para a geração de janelas em alguns casos.

Geração de janelas

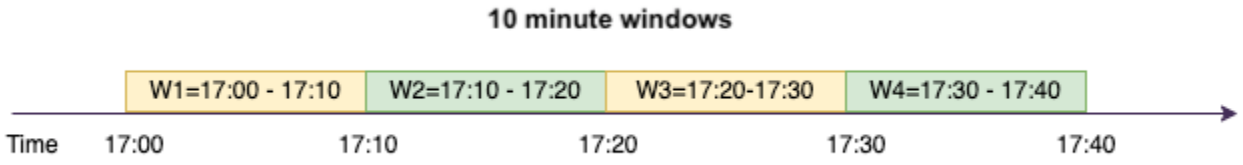
A criação de janelas é uma técnica em que você agrupa e agrega vários eventos. event-time-window Exploraremos, nos exemplos a seguir, os benefícios da geração de janelas e quando você poderia usá-la.

Dependendo do caso de uso de negócio, existem três tipos de janelas de horários com suporte do Spark.

- Janela giratória — uma série de tamanhos fixos não sobrepostos event-time-windows sobre os quais você agrega.
- Janela deslizante: semelhante às janelas em cascata em relação a terem um “tamanho fixo”, mas as janelas podem se sobrepor ou deslizar desde que a duração do deslizamento seja menor que a duração da própria janela.
- Janela de sessão: começa com um evento de dados de entrada e continua a se expandir enquanto recebe entradas dentro de um intervalo ou período de inatividade. Uma janela de sessão pode ter um tamanho estático ou dinâmico da dimensão da janela, dependendo das entradas.

Janela em cascata

A janela giratória é uma série de tamanhos fixos não sobrepostos event-time-windows sobre os quais você agrega. Vamos entender isso com um exemplo real.



A empresa ABC Auto deseja realizar uma campanha de marketing para uma nova marca de carros esportivos. Eles desejam escolher uma cidade na qual haja um grande número de fãs de carros esportivos. Para atingir esse objetivo, a empresa mostra um pequeno anúncio de 15 segundos apresentando o carro em seu site. Todos os “cliques” e a “cidade” correspondente são gravados e transmitidos para. Amazon Kinesis Data Streams Desejamos contabilizar o número de cliques em uma janela de dez minutos e agrupá-los por cidade para visualizar qual cidade tem a maior demanda. Veja a seguir a saída da agregação.

window_start_time	window_end_time	city	total_clicks
2023-07-10 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

Conforme explicado acima, eles event-time-windows são diferentes dos intervalos de tempo de gatilho. Por exemplo, mesmo que o horário do acionamento seja a cada minuto, os resultados de saída mostrarão somente janelas de agregação não sobrepostas de dez minutos. Para otimização, é melhor ter o intervalo do gatilho alinhado com o. event-time-window

Na tabela acima, Dallas realizou 75 cliques na janela das 17h às 17h10, enquanto Chicago realizou dez cliques. Além disso, não há dados para a janela das 17h10 às 17h20 para nenhuma cidade; portanto, essa janela é omitida.

Agora, é possível executar análises aprofundadas nesses dados na aplicação de análise downstream para determinar a cidade mais exclusiva para executar a campanha de marketing.

Uso de janelas em cascata no AWS Glue

1. Crie um Amazon Kinesis Data Streams DataFrame e leia a partir dele. Exemplo:

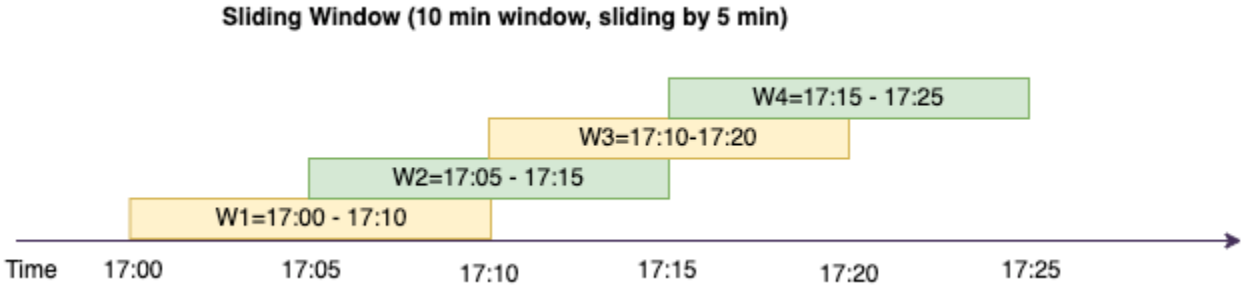
```
parsed_df = kinesis_raw_df \
                .selectExpr('CAST(data AS STRING)') \
                .select(from_json("data", ticker_schema).alias("data")) \
                .select('data.event_time','data.ticker','data.trade','data.volume',
 'data.price')
```

2. Processe dados em uma janela em cascata. No exemplo abaixo, os dados são agrupados com base no campo de entrada “event_time” em janelas em cascata de dez minutos e a saída é gravada em um data lake do 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()
```

Janela deslizante

As janelas deslizantes são semelhantes às janelas em cascata em relação a terem um “tamanho fixo”, mas as janelas podem se sobrepor ou deslizar desde que a duração do deslizamento seja menor que a duração da própria janela. Devido à natureza do deslizamento, uma entrada pode ser vinculada a múltiplas janelas.



Para uma melhor compreensão, vamos considerar o exemplo de um banco que deseja detectar possíveis fraudes relacionadas a cartões de crédito. Uma aplicação de streaming poderia monitorar um fluxo contínuo de transações de cartões de crédito. Essas transações poderiam ser agregadas em janelas de dez minutos de duração e, a cada cinco minutos, a janela avançaria, eliminando os cinco minutos de dados mais antigos e adicionando os cinco minutos de novos dados mais recentes. Dentro de cada janela, as transações poderiam ser agrupadas por país, verificando padrões suspeitos, como uma transação nos Estados Unidos imediatamente seguida por outra na Austrália. Para simplificar, categorizaremos essas transações como fraude quando o valor total das transações for superior a USD 100. Se esse padrão for detectado, uma possível fraude será sinalizada e o cartão poderá ser congelado.

O sistema de processamento de cartões de crédito está enviando uma série de eventos de transação ao Kinesis para cada identificação de cartão em conjunto com o país. Um AWS Glue trabalho executa a análise e produz a seguinte saída agregada.

window_start_time	window_end_time	card_last_four	country	total_amount
2023-07-10 17:00:00	2023-07-10 17:10:00	6544	EUA	85
2023-07-10 17:00:00	2023-07-10 17:10:00	6544	Austrália	10

window_start_time	window_end_time	card_last_four	country	total_amount
2023-07-10 17:05:45	2023-07-10 17:15:45	6544	EUA	50
2023-07-10 17:10:45	2023-07-10 17:20:45	6544	EUA	50
2023-07-10 17:10:45	2023-07-10 17:20:45	6544	Austrália	150

Com base na agregação acima, você pode visualizar a janela de dez minutos deslizando a cada cinco minutos, somada pelo valor da transação. A anomalia é detectada na janela das 17h10 às 17h20, na qual há um ponto fora da curva, que é uma transação de USD 150 na Austrália. O AWS Glue pode detectar essa anomalia e enviar um evento de alarme com a chave do infrator para um tópico do SNS usando o Boto3. Além disso, uma função Lambda pode se inscrever neste tópico e agir.

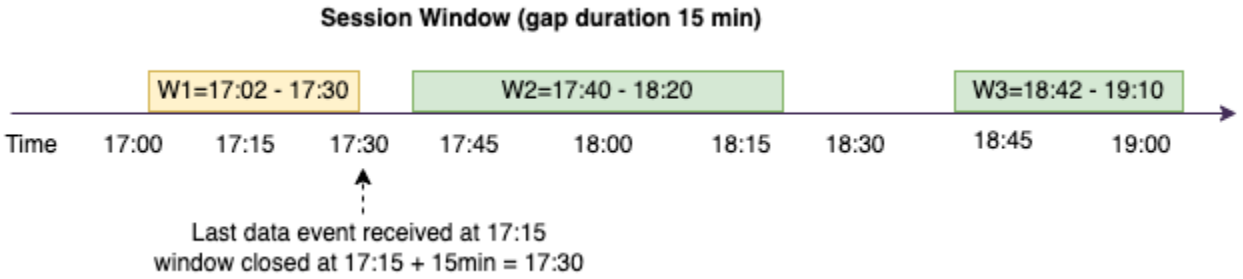
Processamento de dados em uma janela deslizante

A cláusula `group-by` e a função de janela são usadas para implementar a janela deslizante, conforme mostrado abaixo.

```
grouped_df = parsed_df \
            .groupBy(window(col("event_time"), "10 minute", "5 min"), "country",
 "card_last_four") \
            .agg(sum("tx_amount").alias("total_amount"))
```

Janela de sessão

Ao contrário das duas janelas apresentadas acima, que possuem tamanho fixo, a janela de sessão pode ter um tamanho estático ou dinâmico da dimensão da janela com base nas entradas. Uma janela de sessão começa com um evento de dados de entrada e continua a se expandir enquanto recebe entradas dentro de um intervalo ou período de inatividade.



Vamos analisar um exemplo. A empresa ABC Hotel deseja descobrir qual é o horário de maior movimento em uma semana e apresentar melhores ofertas para seus hóspedes. Assim que um convidado faz o check-in, uma janela de sessão é iniciada e o Spark mantém um estado com agregação para isso. event-time-window Toda vez que um convidado faz o check-in, um evento é gerado e enviado para Amazon Kinesis Data Streams. O hotel decide que, se não houver check-in por um período de 15 minutos, ele event-time-window poderá ser fechado. O próximo event-time-window começará novamente quando houver um novo check-in. A saída é semelhante à apresentada a seguir.

window_start_time	window_end_time	city	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

A primeira realização de check-in ocorreu em event_time=17h02. A agregação event-time-window começará às 17:02. Essa agregação continuará enquanto recebermos eventos com intervalos de até 15 minutos. No exemplo acima, o último evento que recebemos foi às 17h15 e, em seguida, nos 15 minutos seguintes não houve eventos. Como resultado, o Spark fechou isso event-time-window às 17:15 +15min = 17:30 e definiu como 17:02 - 17:30. Começou de novo event-time-window às 17:47 quando recebeu um novo evento de dados de check-in.

Processamento de dados em uma janela de sessão

A cláusula `group-by` e a função de janela são usadas para implementar a janela deslizante.

```
grouped_df = parsed_df \
            .groupBy(session_window(col("event_time"), "10 minute"), "city") \
            .agg(count("check_in").alias("total_checkins"))
```

Modos de saída

O modo de saída corresponde ao modo no qual os resultados da tabela ilimitada são gravados no coletor externo. Existem três modos disponíveis. No exemplo apresentado a seguir, você está contabilizando as ocorrências de uma palavra à medida que as linhas de dados são transmitidas e processadas em cada micro lote.

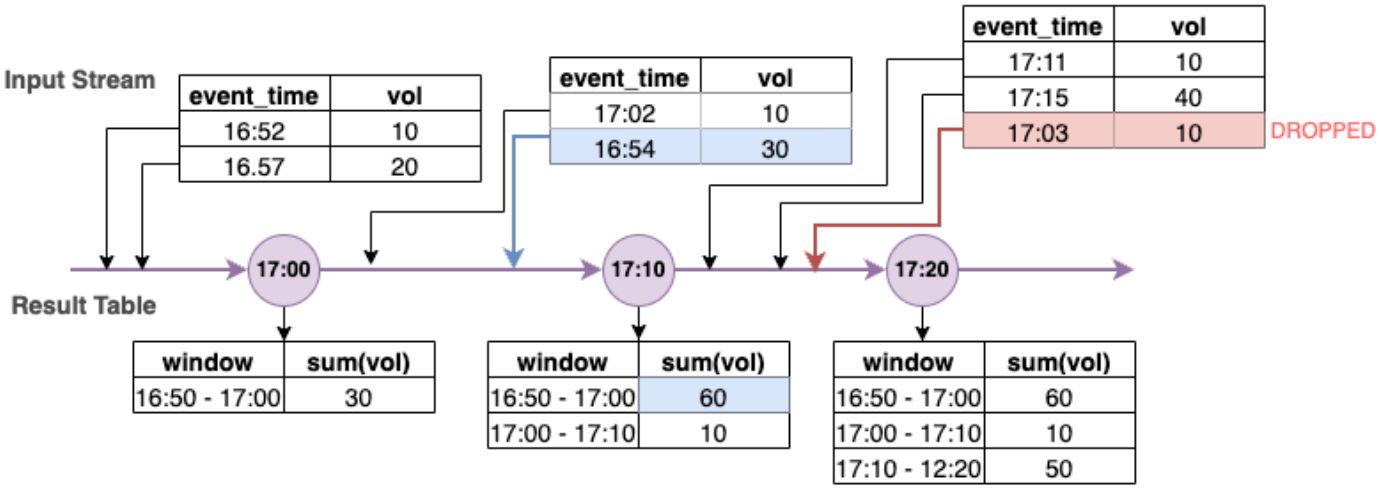
- Modo completo — Toda a tabela de resultados será gravada no coletor após cada processamento em microlote, mesmo que a contagem de palavras não tenha sido atualizada na versão atual event-time-window.
- Modo de acréscimo: esse é o modo padrão, em que somente as novas palavras e as novas linhas adicionadas à tabela de resultados desde o último acionamento serão gravadas no coletor. Esse modo é bom para o streaming sem estado para consultas, por exemplo, as funções map, flatMap, filter etc.
- Modo de atualização: somente as palavras e as linhas na Tabela de Resultados que foram atualizadas ou adicionadas desde o último acionamento serão gravadas no coletor.

> **Note**
>
> O modo de saída = “atualização” não é compatível com as janelas de sessão.

Tratamento de dados atrasados e de marcas d’água

Ao trabalhar com dados em tempo real, pode haver atrasos na chegada dos dados devido à latência da rede e falhas no upstream, e precisamos de um mecanismo para realizar a agregação novamente dos perdidos. event-time-window No entanto, para fazer isso, o estado precisa ser mantido. Ao mesmo tempo, os dados mais antigos precisam ser limpos para limitar o tamanho do estado. A versão 2.1 do Spark adicionou suporte para um recurso, chamado marca d’água, que mantém o estado e permite ao usuário especificar o limite para os dados atrasados.

Com referência ao nosso exemplo de cotações da bolsa acima, consideraremos o limite permitido para os dados atrasados como não superior a dez minutos. Para simplificar, assumiremos uma janela deslizante em que as cotações serão marcadas como AMZ e as negociações como BUY.



No diagrama acima, estamos calculando o volume total em uma janela deslizante de dez minutos. Temos o acionamento às 17h00, 17h10 e 17h20. Acima da seta da linha do tempo, temos o fluxo de dados de entrada e abaixo está a tabela ilimitada de resultados.

Na primeira janela deslizante de dez minutos, realizamos as agregações com base em event_time e o total_volume foi calculado como 30. No segundo event-time-window, o spark obteve o primeiro evento de dados com event_time= 17:02. Como esse é o event_time máximo obtido até o momento pelo Spark, o limite da marca d'água é definido 10 minutos atrás (ou seja, watermark_event_time=16h52). Qualquer evento de dados com um event_time após 16h52 será considerado para a agregação com limite de tempo e qualquer evento de dados anterior será descartado. Isso permite que o Spark mantenha um estado intermediário por mais dez minutos para acomodar dados atrasados. Por volta das 17h08, o Spark recebeu um evento com event_time=16h54, que estava dentro do limite. Portanto, o Spark recalculou as "16:50 - 17:00 " event-time-window e o volume total foi atualizado de 30 para 60.

No entanto, no horário de acionamento 17h20, quando o Spark recebeu o evento com event_time=17h15, ele definiu watermark_event_time=17h05. Como consequência, o evento de dados atrasado com event_time=17h03 foi considerado "tarde demais" e ignorado.

```
Watermark Boundary = Max(Event Time) - Watermark Threshold
```

Uso de marcas d'água no AWS Glue

O Spark só emitirá ou gravará os dados no coletor externo quando o limite da marca d'água for ultrapassado. Para implementar uma marca d'água no AWS Glue, consulte o exemplo abaixo.

```
grouped_df = parsed_df \
                .withWatermark("event_time", "10 minutes") \
                .groupBy(window("event_time", "5 minutes"), "ticker") \
                .agg(sum("volume").alias("total_volume"))
```

Monitoramento de trabalhos de streaming do AWS Glue

O monitoramento do trabalho de streaming corresponde a uma parte crítica do desenvolvimento do pipeline de ETL. Além de usar a interface de usuário do Spark, você também pode usar o Amazon CloudWatch para monitorar as métricas. Abaixo é apresentada uma lista de métricas de streaming emitidas pela estrutura do AWS Glue. Para obter uma lista completa de todas as métricas do AWS Glue, consulte Monitorar o AWS Glue usando métricas do Amazon CloudWatch.

O AWS Glue usa uma estrutura de streaming articulada para processar os eventos de entrada. É possível usar a API do Spark diretamente em seu código ou aproveitar o `ForEachBatch` fornecido por `GlueContext`, que publica essas métricas. Para compreender essas métricas, primeiro, é necessário compreender `windowSize`.

windowSize: `windowSize` corresponde ao intervalo de micro lote que você fornece. Se você especificar um tamanho da janela de 60 segundos, o trabalho de streaming do AWS Glue aguardará 60 segundos (ou mais, se o lote anterior ainda não tiver sido concluído) antes de ler os dados em um lote da fonte de streaming e aplicar as transformações fornecidas em `ForEachBatch`. Isso também é conhecido como intervalo de acionamento.

Analisaremos as métricas com mais detalhes para compreender as características de integridade e de performance.

> **Note**
>
> As métricas são emitidas a cada 30 segundos. Se o `windowSize` tiver menos de 30 segundos, as métricas relatadas corresponderão a uma agregação. Por exemplo, suponhamos que o `windowSize` tenha dez segundos e você esteja processando

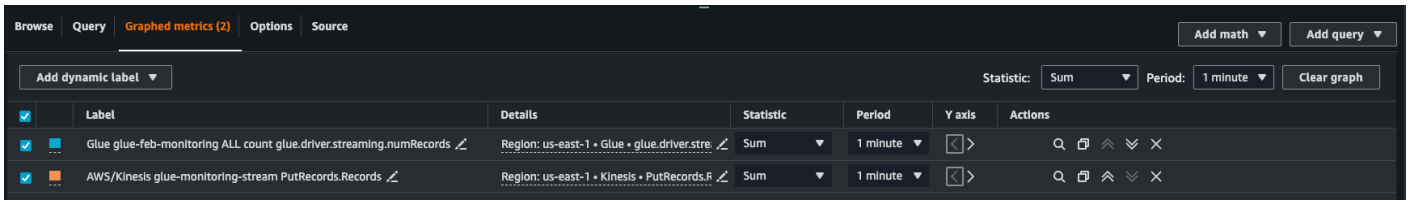
20 registros por micro lote de forma contínua. Neste cenário, o valor da métrica emitida para numRecords seria 60.
Uma métrica não será emitida se não houver dados disponíveis para ela. Além disso, no caso de uma métrica de atraso do consumidor, é necessário habilitar o recurso para obter métricas para ela.

Visualização de métricas

Para traçar métricas visuais:

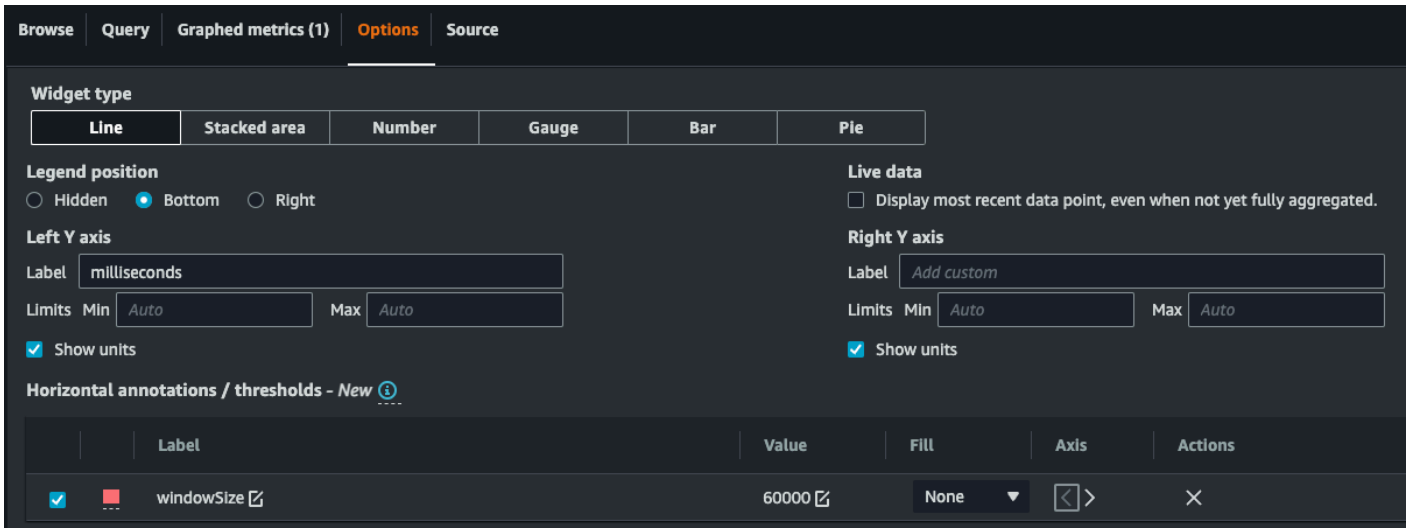
1. Acesse Métricas no console do Amazon CloudWatch e, em seguida, escolha a guia Procurar. Em seguida, selecione Glue em “Namespaces personalizados”.

2. Escolha Métricas de trabalho para que as métricas de todos os seus trabalhos sejam apresentadas.
3. Filtre as métricas com base em JobName=glue-feb-monitoring e, em seguida, em JobRunId=ALL. É possível clicar no sinal “+”, conforme mostrado na figura abaixo, para fazer adições ao filtro de pesquisa.
4. Selecione a caixa de seleção das métricas nas quais você tem interesse. Na figura abaixo, selecionamos `numberAllExecutors` e `numberMaxNeededExecutors`.

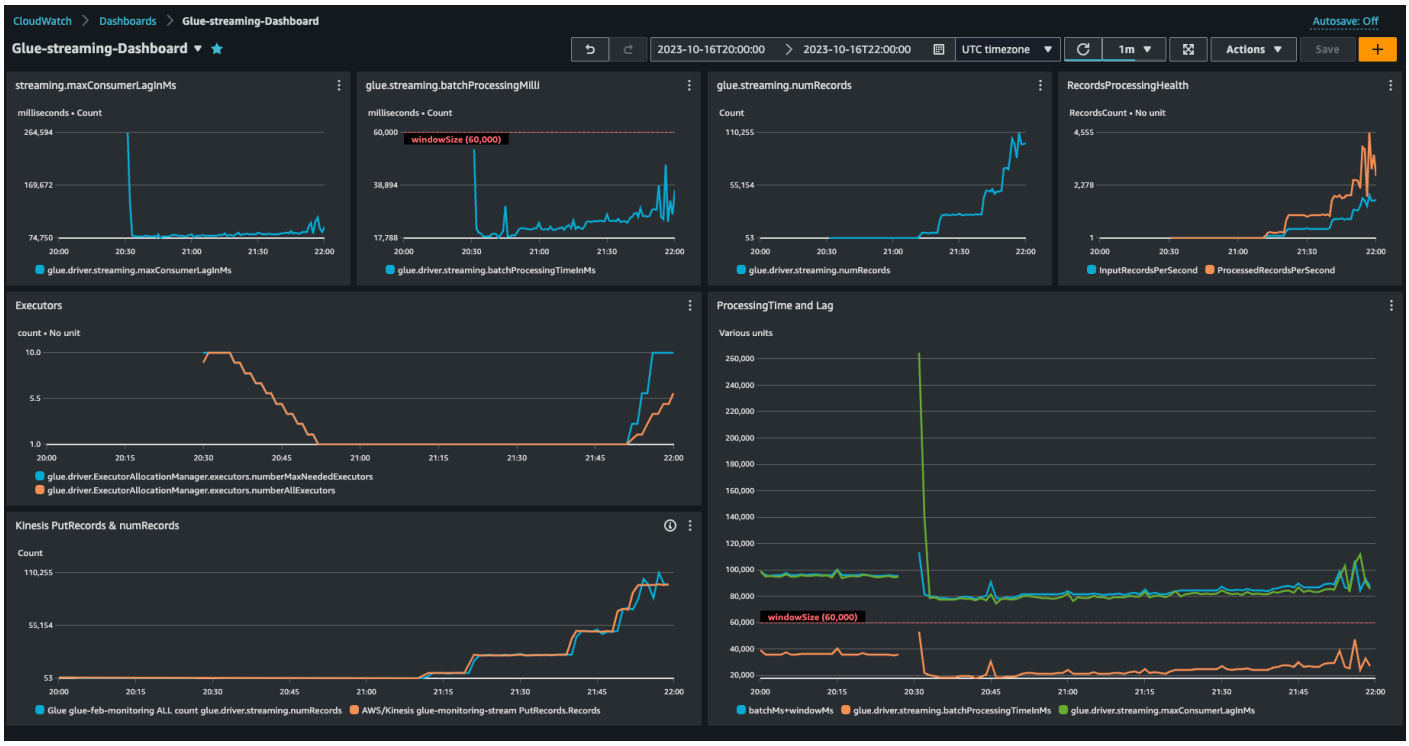


5. Depois de selecionar essas métricas, é possível acessar a guia Métricas em gráficos e aplicar suas estatísticas.
6. Como as métricas são emitidas a cada minuto, você pode aplicar a “média” ao longo de um minuto para `batchProcessingTimeInMs` e `maxConsumerLagInMs`. Para `numRecords`, é possível aplicar a “soma” ao longo de cada minuto.

7. Você pode adicionar uma anotação horizontal `windowSize` à sua representação em gráfico usando a guia Opções.



8. Depois de selecionar as métricas, crie um painel e realize as adições. Veja a seguir um painel de amostra.

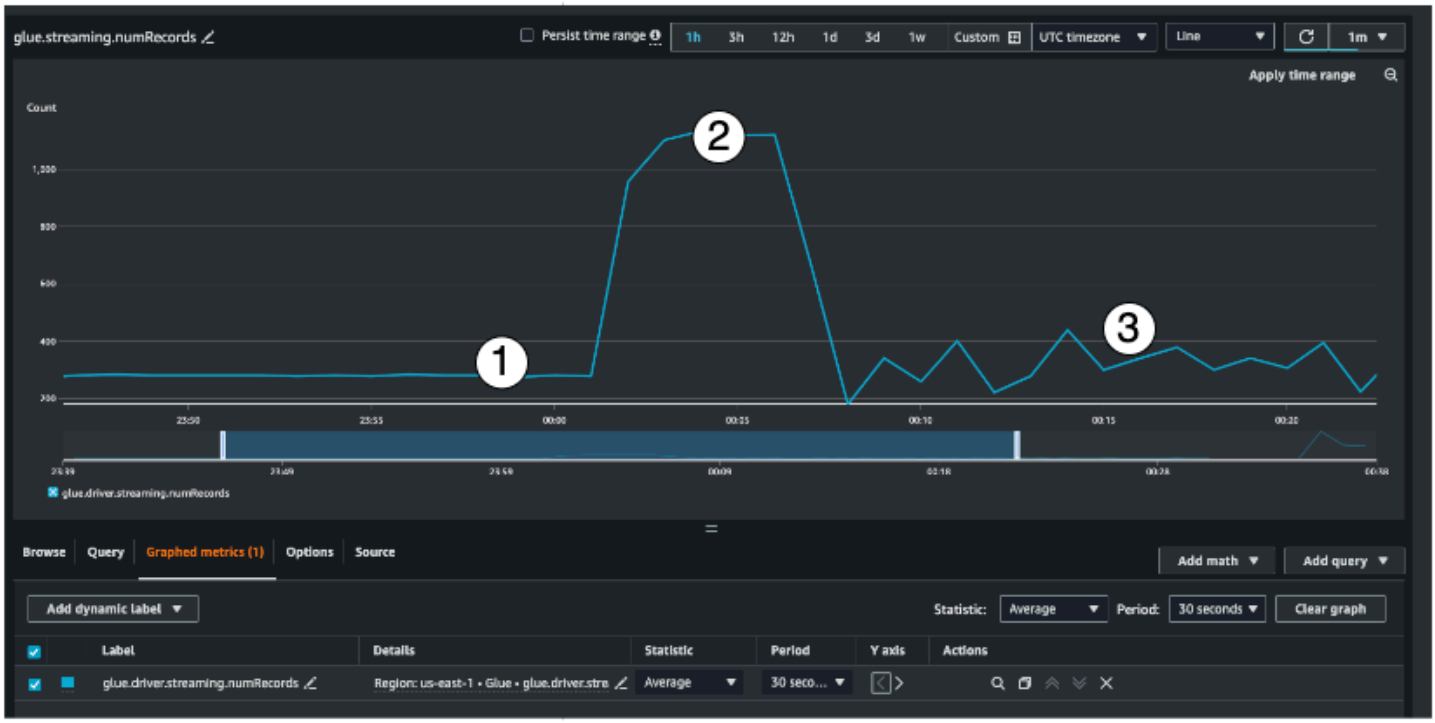


Aprofundamento das métricas

Essa seção descreve cada uma das métricas e como elas se correlacionam entre si.

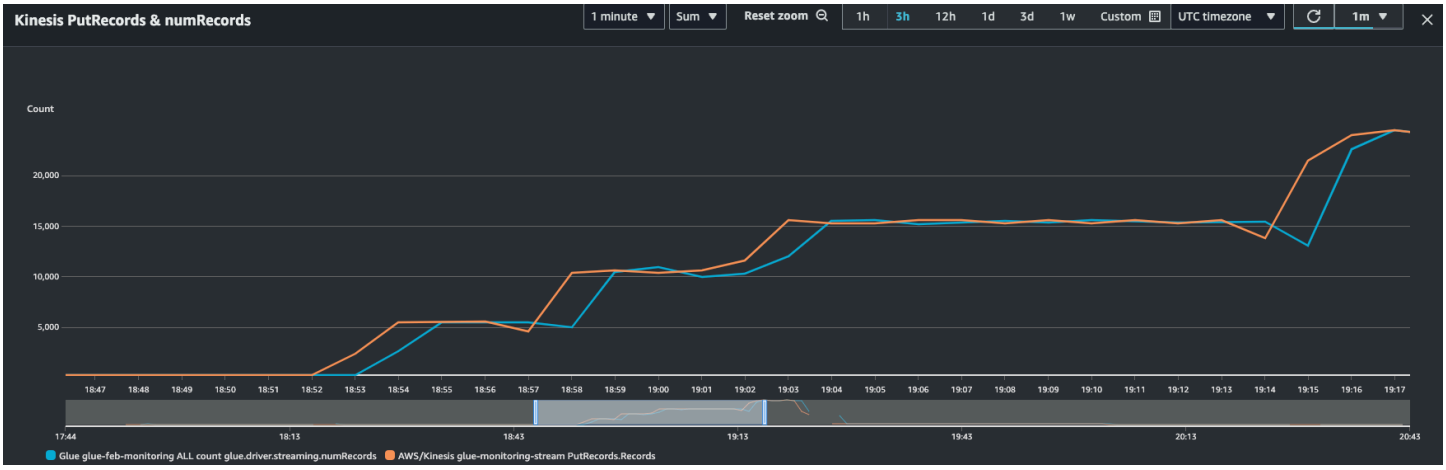
Número de registros (métrica: streaming.numRecords)

Essa métrica indica quantos registros estão sendo processados.



Essa métrica de streaming fornece visibilidade sobre o número de registros que você está processando em uma janela. Além do número de registros que estão sendo processados, ela ajudará você a compreender o comportamento do tráfego de entrada.

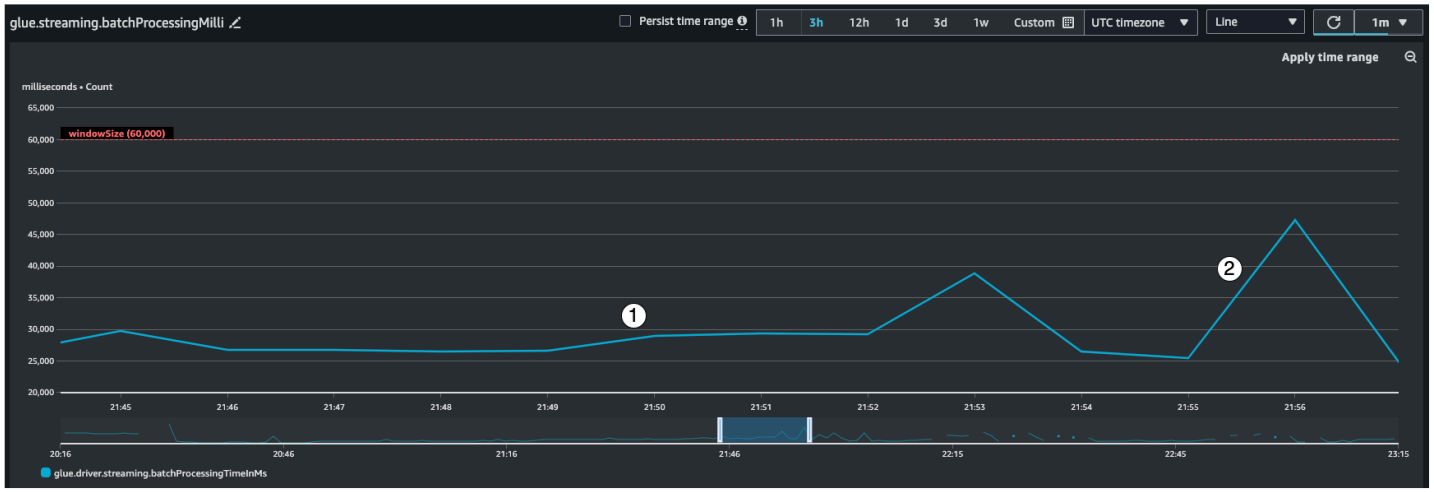
- O indicador n.º 1 mostra um exemplo de tráfego estável sem aumentos. Normalmente, serão aplicações, como sensores de IoT, que coletam dados em intervalos regulares e os enviam para a fonte de streaming.
- O indicador n.º 2 mostra um exemplo de um aumento repentino no tráfego em uma carga estável. Isso pode acontecer em uma aplicação de fluxo de cliques quando há um evento de marketing, como a Black Friday, e ocorre um aumento no número de cliques.
- O indicador n.º 3 mostra um exemplo de tráfego imprevisível. O tráfego imprevisível significa que há um problema. É somente a natureza dos dados de entrada. Ao retornarmos ao exemplo do sensor de IoT, é possível imaginar centenas de sensores que enviam eventos de mudanças climáticas para a fonte de streaming. Como a mudança climática não é previsível, os dados também não o são. A compreensão do padrão de tráfego é fundamental para dimensionar seus executores. Se a entrada tiver um amplo aumento, considere usar o ajuste de escala automático (falaremos mais sobre isso posteriormente).



É possível combinar essa métrica com a métrica PutRecords do Kinesis para garantir que o número de eventos que estão sendo ingeridos e o número de registros que estão sendo lidos sejam praticamente semelhantes. Isso é especialmente útil quando você está tentando compreender o atraso. À medida que a taxa de ingestão aumenta, o mesmo acontece com os `numRecords` lidos pelo AWS Glue.

Tempo de processamento em lote (métrica: streaming.batchProcessingTimeInMs)

A métrica de tempo de processamento em lote ajuda a determinar se o cluster está com provisionamento insuficiente ou excessivo.

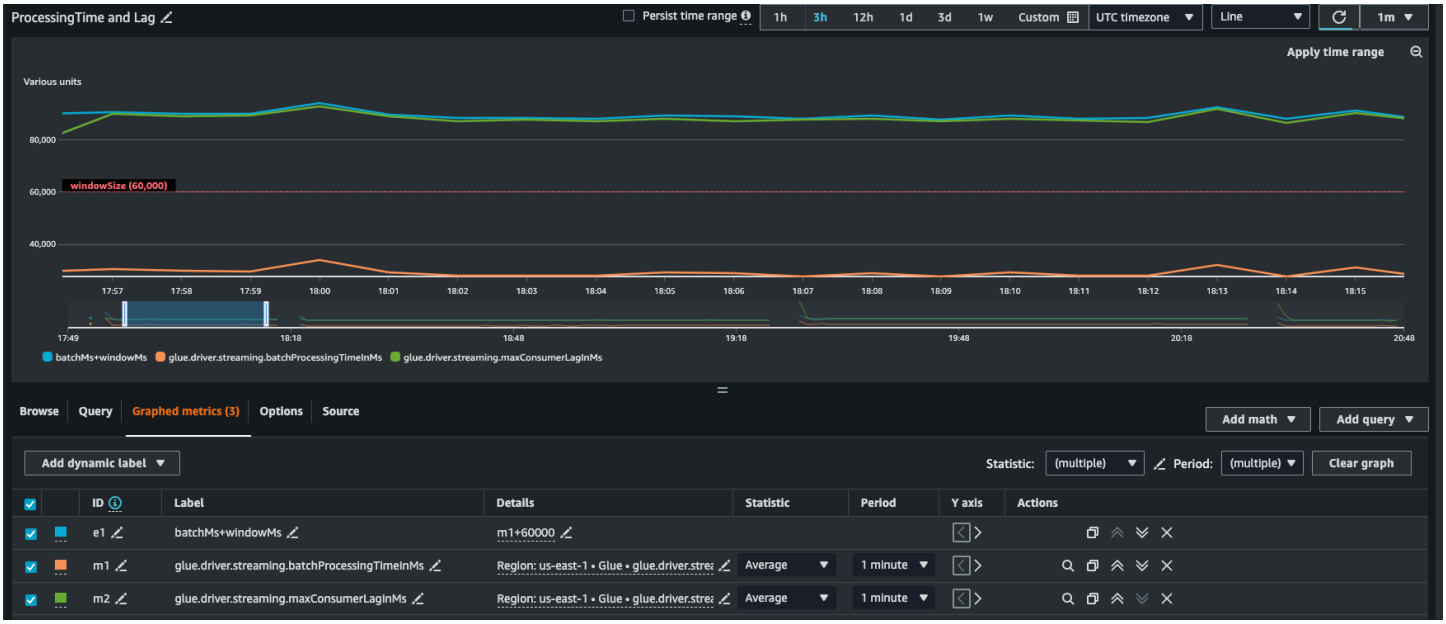


Essa métrica indica o número de milissegundos necessários para processar cada micro lote de registros. Aqui, o principal objetivo é monitorar esse tempo para garantir que seja menor que o intervalo de `windowSize`. Não há problema se `batchProcessingTimeInMs` for temporariamente desativado, desde que se recupere no intervalo de janela seguinte. O indicador n.º 1 mostra um tempo mais ou menos estável necessário para o processamento do trabalho. No entanto, se o

número de registros de entrada estiver aumentando, o tempo necessário para o processamento do trabalho aumentará, conforme mostrado pelo indicador n.º 2. Se o `numRecords` não estiver aumentando, mas o tempo de processamento estiver, você precisará examinar o processamento do trabalho nos executores de forma mais aprofundada. É uma boa prática definir um limite e um alarme para garantir que `batchProcessingTimeInMs` não ultrapasse 120% por mais de dez minutos. Para obter mais informações sobre como configurar alarmes, consulte Usar alarmes do Amazon CloudWatch.

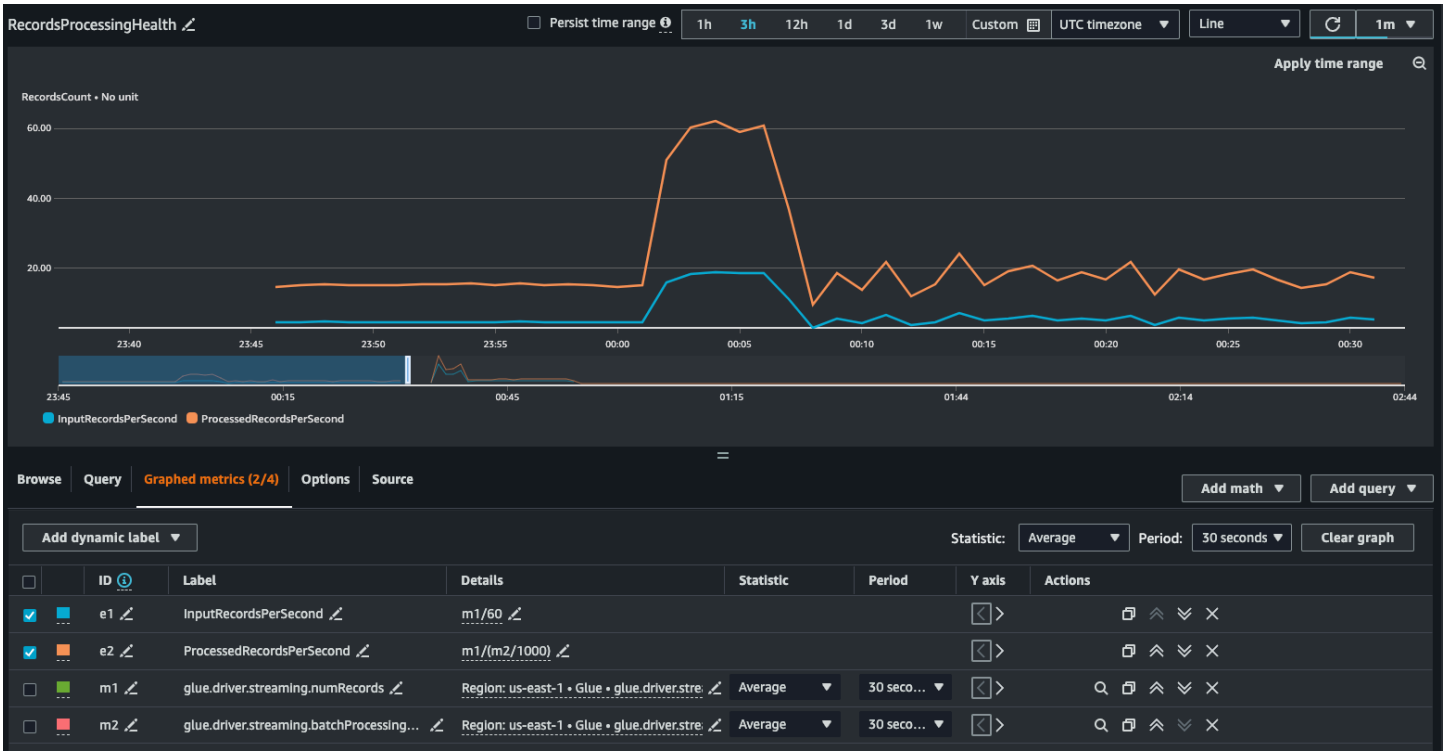
Atraso do consumidor (métrica: streaming.maxConsumerLagInMs)

A métrica de atraso do consumidor ajuda a entender se há um atraso no processamento de eventos. Se o atraso for muito elevado, você poderá perder o SLA de processamento do qual sua empresa depende, mesmo que tenha um windowSize correto. Você deve habilitar explicitamente essas métricas usando a opção de conexão `emitConsumerLagMetrics`. Para obter mais informações, consulte KinesisStreamingSourceOptions.



Métricas derivadas

Para obter insights mais aprofundados, é possível criar métricas derivadas para compreender mais sobre os trabalhos de streaming no Amazon CloudWatch.



É possível desenvolver uma representação em gráfico com métricas derivadas para decidir se precisa usar mais DPUs. Embora o ajuste de escala automático ajude você a fazer isso automaticamente, é possível usar métricas derivadas para determinar se o ajuste de escala automático está funcionando de maneira eficaz.

- InputRecordsPerSecond indica a taxa na qual você está obtendo registros de entrada. A derivação ocorre da seguinte forma: número de registros de entrada (glue.driver.streaming.numRecords)/ WindowSize.
- ProcessingRecordsPerSecond indica a taxa na qual os registros estão sendo processados. A derivação ocorre da seguinte forma: número de registros de entrada (glue.driver.streaming.numRecords)/batchProcessingTimeInMs.

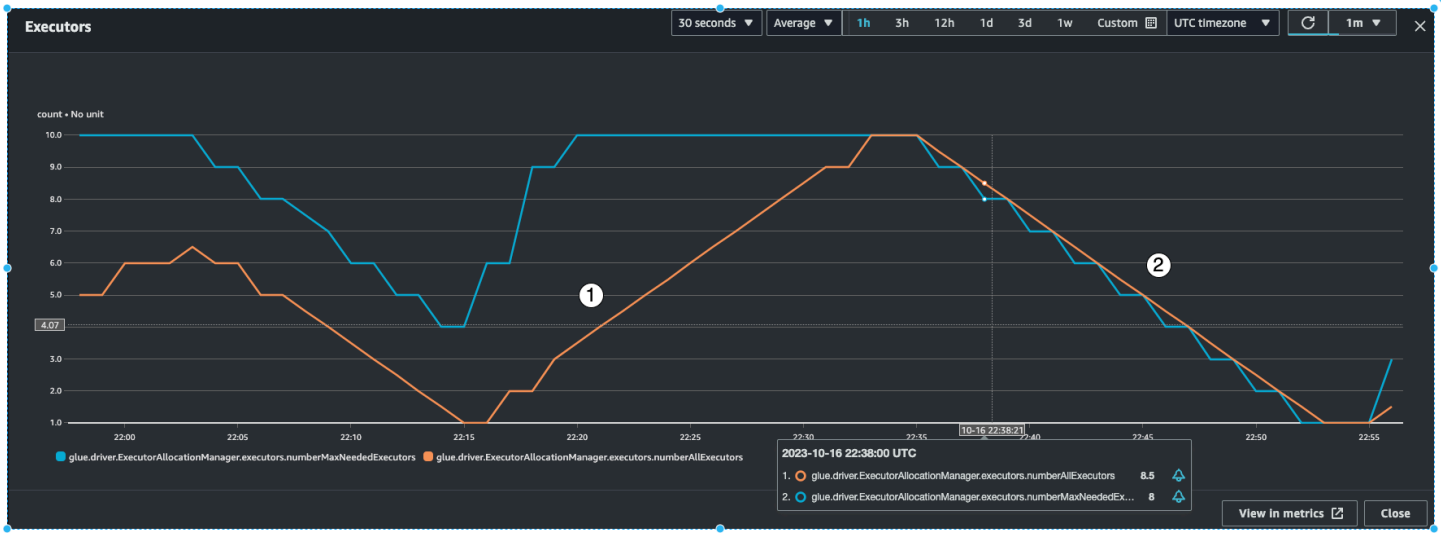
Se a taxa de entrada for superior à taxa de processamento, talvez seja necessário adicionar mais capacidade para processar os trabalhos ou aumentar o paralelismo.

Métricas do ajuste de escala automático

Quando o tráfego de entrada tiver um amplo aumento, considere habilitar o ajuste de escala automático e especificar o número máximo de trabalhadores. Com isso você obtém duas métricas adicionais, `numberAllExecutors` e `numberMaxNeededExecutors`.

- numberAllExecutors corresponde ao número de executores de trabalhos em execução ativa.
- numberMaxNeededExecutors corresponde ao número máximo de executores de trabalhos (em execução ativa e pendentes) necessários para satisfazer a carga atual.

Essas duas métricas ajudarão você a entender se o ajuste de escala automático está funcionando corretamente.



O AWS Glue monitorará a métrica `batchProcessingTimeInMs` em alguns micro lotes e fará uma entre duas coisas. Ele aumentará a escala horizontalmente dos executores, se `batchProcessingTimeInMs` estiver mais próximo de `windowSize`, ou reduzirá a escala horizontalmente dos executores, se `batchProcessingTimeInMs` for comparativamente menor que `windowSize`. Além disso, ele usará um algoritmo para escalar os executores em etapas.

- O indicador n.º 1 mostra como os executores ativos aumentaram a escala verticalmente para alcançar o número máximo de executores necessários para o processamento da carga.
- O indicador n.º 2 mostra como os executores ativos reduziram a escala horizontalmente desde que `batchProcessingTimeInMs` estava baixo.

É possível usar essas métricas para monitorar o paralelismo atual no nível do executor e ajustar, adequadamente, o número máximo de trabalhadores em sua configuração de ajuste de escala automático.

Como obter a melhor performance

O Spark tentará criar uma tarefa por fragmento para leitura no fluxo do Amazon Kinesis. Os dados em cada fragmento se tornam uma partição. Em seguida, ele distribuirá essas tarefas entre os executores e os trabalhadores, dependendo do número de núcleos em cada trabalhador (o número de núcleos por trabalhador depende do tipo de trabalhador que você selecionar, por exemplo, `G.025X`, `G.1X`, entre outros). No entanto, não é determinística a forma como as tarefas são distribuídas. Todas as tarefas são executadas em paralelo em seus respectivos núcleos. As tarefas serão enfileiradas se houver mais fragmentos do que o número de núcleos executores disponíveis.

É possível usar uma combinação das métricas acima e do número de fragmentos para provisionar os executores para uma carga estável com algum espaço para aumentos. É recomendável executar algumas iterações do seu trabalho para determinar o número aproximado de trabalhadores. Para uma workload instável ou com amplo aumento, é possível fazer o mesmo ao configurar o ajuste de escala automático e o número máximo de trabalhadores.

Defina o `windowSize` de acordo com os requisitos de SLA da sua empresa. Por exemplo, se sua empresa exigir que os dados processados não podem ficar obsoletos por mais de 120 segundos, defina `windowSize` para, no mínimo, 60 segundos, de modo que o atraso médio do consumidor seja inferior a 120 segundos (consulte a seção sobre atraso do consumidor acima). A partir desse ponto, dependendo dos `numRecords` e do número de fragmentos, planeje a capacidade em DPUs, certificando-se de que `batchProcessingTimeInMs` seja inferior a 70% do `windowSize` na maioria das vezes.

> **Note**
>
> Os fragmentos ativos podem causar distorção de dados, o que significa que alguns fragmentos e partições são muito maiores do que outros. Isso pode fazer com que algumas tarefas executadas em paralelo demorem mais, causando tarefas retardatárias. Como resultado, o próximo lote só poderá ser iniciado quando todas as tarefas do anterior forem concluídas. Isso afetará o `batchProcessingTimeInMillis` e o atraso máximo.

AWS Glue Qualidade de dados

AWS Glue A qualidade dos dados permite medir e monitorar a qualidade dos seus dados para que você possa tomar boas decisões comerciais. Construído com base na DeeQu estrutura de código aberto, o AWS Glue Data Quality fornece uma experiência gerenciada e sem servidor. AWS Glue A Qualidade de Dados funciona com a Linguagem de Definição de Qualidade de Dados (DQDL), que é uma linguagem específica de domínio que você usa para definir regras de qualidade de dados. Para saber mais sobre a DQDL e os tipos de regras compatíveis, consulte Referência de Data Quality Definition Language (DQDL).

Para detalhes adicionais sobre o produto e os preços, consulte a página de serviços do AWS Glue Data Quality.

Benefícios e principais atributos

Os benefícios e os principais recursos da qualidade de AWS Glue dados incluem:

- Tecnologia sem servidor: não requer instalação, correção nem manutenção.
- Comece rapidamente — o AWS Glue Data Quality analisa rapidamente seus dados e cria regras de qualidade de dados para você. Você pode começar com dois cliques: “Criar regras de qualidade de dados → Recomendar regras”.
- Detecte problemas de qualidade de dados — Use o aprendizado de máquina (ML) para detectar anomalias e problemas de qualidade hard-to-detect de dados.
- Improvise suas regras — com mais de 25 regras de out-of-the-box DQ para começar, você pode criar regras que atendam às suas necessidades específicas.
- Avalie a qualidade e tome decisões comerciais confiáveis: depois de avaliar as regras, você obtém uma pontuação de qualidade de dados que fornece uma visão geral da integridade dos dados. Use a pontuação de qualidade de dados para tomar decisões de negócios confiáveis.
- Concentre-se em dados incorretos — A qualidade de AWS Glue dados ajuda você a identificar os registros exatos que causaram a queda de seus índices de qualidade. Identifique-os facilmente, coloque-os em quarentena e corrija-os.
- Pague conforme o uso — Não há licenças anuais necessárias para usar o AWS Glue Data Quality.
- Sem restrições — o AWS Glue Data Quality é baseado em código aberto DeeQu, permitindo que você mantenha as regras que você está criando em uma linguagem aberta.

- Verificações de qualidade de AWS Glue dados — Qualidade de dados Você pode aplicar verificações de qualidade de dados em Data Catalog pipelines de AWS Glue ETL, permitindo gerenciar a qualidade dos dados em repouso e em trânsito.
- Detecção de qualidade de dados baseada em ML — Use o aprendizado de máquina (ML) para detectar anomalias e hard-to-detect problemas de qualidade de dados.

Como funciona

Há dois pontos de entrada para qualidade AWS Glue de dados: as tarefas AWS Glue Data Catalog e AWS Glue ETL. Esta seção fornece uma visão geral dos casos de uso e dos AWS Glue recursos que cada ponto de entrada suporta.

Qualidade de dados para o AWS Glue Data Catalog

AWS Glue A qualidade de dados avalia objetos armazenados no. AWS Glue Data Catalog Ela oferece aos não codificadores uma maneira fácil de configurar regras de qualidade de dados. Essas pessoas incluem administradores de dados e analistas de negócios.

Você pode escolher essa opção para os seguintes casos de uso:

- Você deseja realizar tarefas de qualidade de dados em conjuntos de dados que já catalogou no AWS Glue Data Catalog.
- Você trabalha com governança de dados e precisa identificar ou avaliar problemas de qualidade de dados no data lake de forma constante.

Você pode gerenciar a qualidade dos dados do catálogo de dados usando as seguintes interfaces:

- O console AWS Glue de gerenciamento
- AWS Glue APIs

Para começar com AWS Glue Data Quality for the AWS Glue Data Catalog seeIntrodução ao AWS Glue Data Quality para o Data Catalog.

Qualidade de dados para trabalhos AWS Glue de ETL

AWS Glue A qualidade de dados para trabalhos de AWS Glue ETL permite que você execute tarefas proativas de qualidade de dados. As tarefas proativas ajudam você a identificar e filtrar dados insatisfatórios antes de carregar um conjunto de dados no data lake.

Vídeo: Apresentando a qualidade AWS Glue de dados para pipelines de ETL

Você pode escolher a qualidade dos dados para trabalhos de ETL para os seguintes casos de uso:

- Você deseja incorporar tarefas de qualidade de dados nos trabalhos de ETL
- Você deseja escrever um código que defina tarefas de qualidade de dados em scripts de ETL
- Você quer gerenciar a qualidade dos dados que fluem pelos pipelines de dados visuais

Você pode gerenciar a qualidade dos dados para trabalhos de ETL usando as seguintes interfaces:

- AWS Glue Studio, AWS Glue Studio cadernos e sessões AWS Glue interativas
- AWS Glue bibliotecas para scripts ETL
- AWS Glue APIs

Para começar com a qualidade de dados para trabalhos de ETL, consulte Tutorial: Getting started with Data Quality no AWS Glue Studio User Guide.

Comparar a qualidade dos dados do catálogo de dados com a qualidade dos dados dos trabalhos de ETL

Esta tabela fornece uma visão geral dos recursos que cada ponto de entrada da Qualidade de AWS Glue Dados suporta.

Atributo	Qualidade dos dados para o catálogo de dados	Qualidade de dados para trabalhos de ETL
Fontes de dados	Amazon S3, Amazon Redshift, fontes JDBC compatíveis com o catálogo de dados e formatos de data lakes transacionais, como Apache	Todas as fontes de dados suportadas pela AWS Glue, incluindo conectores personalizados e conectores de terceiros.

Atributo	Qualidade dos dados para o catálogo de dados	Qualidade de dados para trabalhos de ETL
	Iceberg, Apache Hudi e Delta Lake. Observe que, se as tabelas forem AWS Lake Formation gerenciadas, as tabelas Iceberg, Delta e HUDI não serão suportadas. Amazon Athena as visualizações que estão catalogadas em não AWS Glue Data Catalog são suportadas.	
Recomendações de regras de qualidade de dados	Compatível	Sem compatibilidade
Criar e executar regras DQDL	Compatível	Compatível
Ajuste de escala automático	Não compatível	Compatível
AWS Glue Suporte Flex	Não compatível	Compatível
Programação	Compatível ao avaliar regras de qualidade de dados e por meio do Step Functions.	Compatível com o uso do Step Functions e fluxos de trabalho.
Identificação de registros que falharam nas verificações de qualidade de dados	Não compatível	Compatível
Integração com o Amazon EventBridge	Compatível	Compatível
Integração com o AWS Cloudwatch	Compatível	Compatível

Atributo	Qualidade dos dados para o catálogo de dados	Qualidade de dados para trabalhos de ETL
Gravar resultados de qualidade de dados no Amazon S3	Compatível	Compatível
Qualidade incremental dos dados	Compatível por meio dos predicados de pushdown	Compatível por meio de AWS Glue favoritos
AWS CloudFormation apoio	Compatível	Compatível
Detecção de anomalias baseada em ML	Sem compatibilidade	Demonstração
Regras dinâmicas	Não compatível	Compatível

Considerações

Considere os seguintes itens antes de usar a Qualidade de AWS Glue dados:

- As regras de qualidade de dados não podem avaliar fontes de dados aninhadas ou do tipo lista. Consulte Nivelar structs aninhados.

Terminologia

A lista a seguir define termos relacionados à qualidade AWS Glue dos dados.

Data Quality Definition Language (DQDL)

Uma linguagem específica do domínio que você pode usar para escrever regras de qualidade de AWS Glue dados.

Para saber mais sobre DQDL, consulte o guia de Referência de Data Quality Definition Language (DQDL).

qualidade de dados

Descreve o quão bem um conjunto de dados atende a sua finalidade específica. AWS Glue A qualidade de dados avalia as regras em relação a um conjunto de dados para medir a qualidade dos dados. Cada regra verifica características específicas, como atualidade ou integridade dos dados. Para quantificar a qualidade dos dados, você pode usar uma pontuação de qualidade de dados.

pontuação de qualidade e dados

A porcentagem de regras de qualidade de dados aprovadas (resultam em verdadeiras) quando você avalia um conjunto de regras com Qualidade de AWS Glue dados.

regra

Uma expressão DQDL que verifica os dados em busca de uma característica específica e retorna um valor booliano. Para ter mais informações, consulte Estrutura da regra.

analisador

Uma expressão de DQDL que reúne estatísticas de dados. Um analisador reúne estatísticas de dados que podem ser usadas por algoritmos de ML para detectar anomalias e problemas de qualidade de hard-to-detect dados ao longo do tempo.

conjunto de regras

Um AWS Glue recurso que compreende um conjunto de regras de qualidade de dados. Um conjunto de regras deve estar associado a uma tabela no AWS Glue Data Catalog. Ao salvar um conjunto de regras, o AWS Glue atribui nome do recurso da Amazon (ARN) ao conjunto de regras.

pontuação de qualidade e dados

A porcentagem de regras de qualidade de dados aprovadas (resultam em verdadeiras) quando você avalia um conjunto de regras com o AWS Glue Data Quality.

observação

Um insight não confirmado gerado pelo AWS Glue pela análise de estatísticas de dados coletadas de regras e analisadores ao longo do tempo.

Limites

AWS Glue Limites do serviço de qualidade de dados:

- Você pode ter 2000 regras em um conjunto de regras. Se seus conjuntos de regras forem maiores, recomendamos dividi-los em vários conjuntos de regras.
- O tamanho do conjunto de regras é 65 KB. Se seus conjuntos de regras forem maiores, recomendamos dividi-los em vários conjuntos de regras.

Notas de lançamento do AWS Glue Data Quality

Este tópico descreve os recursos introduzidos na Qualidade de AWS Glue dados.

Disponibilidade geral: novos atributos

Os seguintes novos recursos estão disponíveis com a disponibilidade geral da Qualidade de AWS Glue Dados:

- A capacidade de identificar quais registros falharam nas verificações de qualidade de dados agora é suportada no AWS Glue Studio
- Novos tipos de regras de qualidade de dados, como validação da integridade referencial de dados entre dois conjuntos de dados, comparação de dados entre dois conjuntos de dados e verificações de tipos de dados
- Experiência de usuário aprimorada no AWS Glue Data Catalog
- Compatibilidade com o Apache Iceberg, o Apache Hudi e o Delta Lake
- Compatibilidade com o Amazon Redshift
- Notificação simplificada com a Amazon EventBridge
- AWS CloudFormation suporte para criar conjuntos de regras
- Melhorias no desempenho: opção de armazenamento em cache no ETL e AWS Glue Studio para um desempenho mais rápido ao avaliar a qualidade dos dados

27 de novembro de 2023 (pré-visualização)

- Os recursos de detecção de anomalias com tecnologia ML agora estão disponíveis no AWS Glue ETL e AWS Glue Studio. Com isso, agora você pode detectar anomalias e problemas de qualidade hard-to-detect de dados.
- As regras dinâmicas permitem que você forneça limites dinâmicos (p. ex.,: `RowCount> avg(last(10))`).

12 de março de 2024

- Melhorias em DQDL
  - Suporte a palavras-chave como NULL, BLANKS, WHITESPACES_ONLY
  - Opções para especificar como a qualidade AWS Glue de dados deve lidar com regras compostas
  - ColumnValues o tipo de regra não permitirá que valores NULL passem durante as comparações
  - Suporte ao operador NOT em DQDL

26 de junho de 2024

- Melhorias em DQDL
  - O DQDL agora suporta a cláusula where para que você possa filtrar dados antes de aplicar as regras do DQ

Detecção de anomalias no AWS Glue Data Quality

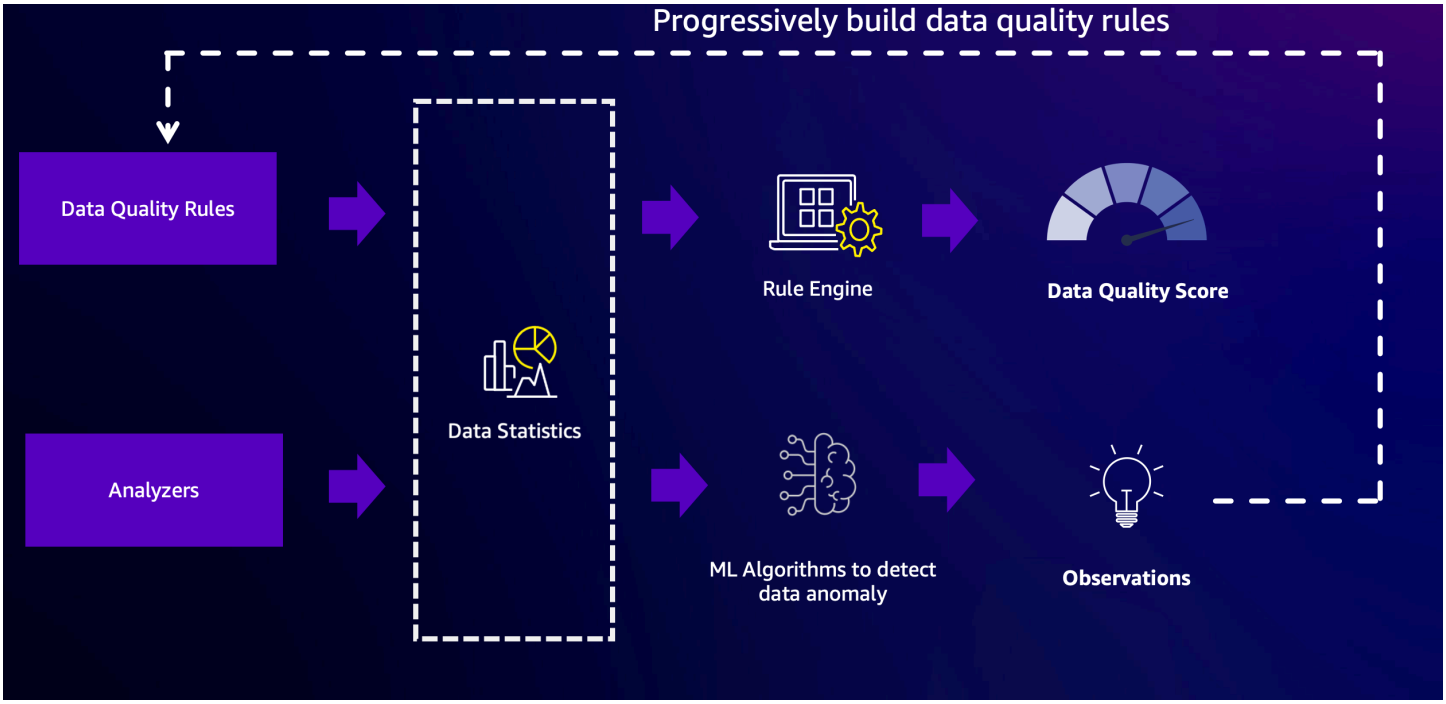
> **Note**
>
> O AWS Glue Data Quality está disponível em versão de pré-lançamento nas seguintes regiões:
>
> - Leste dos EUA (Ohio, Norte da Virgínia)
> - Oeste dos EUA (Oregon)
> - Ásia-Pacífico (Tóquio)
> - Europa (Irlanda)

A detecção de anomalias no AWS Glue Data Quality usa algoritmos de machine learning (ML) nas estatísticas de dados ao longo do tempo para detectar padrões anormais e problemas ocultos de qualidade de dados que são difíceis de detectar por meio de regras. Até a presente data, a detecção de anomalias está disponível apenas para a versão 4.0 do AWS Glue. Esse recurso está atualmente disponível apenas para o AWS Glue Studio Visual ETL e AWS Glue ETL. Esse recurso não funciona nos cadernos do AWS Glue Studio, no catálogo de dados do AWS Glue, nas sessões interativas do AWS Glue nem nas visualizações de dados do AWS Glue.

Como funciona

Ao avaliar as regras do Data Quality, o AWS Glue registra as estatísticas de dados necessárias para determinar se os dados estão em conformidade com as regras. Por exemplo, o Data Quality calculará o número de valores distintos em um conjunto de dados e, em seguida, comparará esse valor com a expectativa.

O mecanismo de regras do Data Quality compara o valor estatístico com os limites definidos e avalia seus requisitos de qualidade. Como essas estatísticas são coletadas ao longo do tempo, você pode habilitar a detecção de anomalias em seus pipelines ETL para fazer com que o AWS Glue aprenda com estatísticas anteriores e reporte padrões ocultos como observações. As observações são insights não confirmados que o algoritmo de ML do AWS Glue identifica. Elas vêm com regras recomendadas do Data Quality que você pode aplicar ao seu conjunto de regras para monitorar o padrão descoberto. Recomendamos executar os trabalhos em uma programação regular (por exemplo, de hora em hora e diariamente). Execuções irregulares podem produzir insights insatisfatórios.



Como usar analisadores para inspecionar dados

Às vezes, você pode não ter tempo para criar regras de qualidade de dados. É aí que os analisadores são úteis. Os analisadores fazem parte do seu conjunto de regras e são muito simples de configurar. Por exemplo, você pode gravar isso em seu conjunto de regras:

```
Analyzers = [
      RowCount,
      Completeness “AllColumns”
]
```

Isso reunirá as seguintes estatísticas:

- Contagem de linhas para todo o conjunto de dados;
- Completude de cada coluna em seu conjunto de dados.

Recomendamos o uso de analisadores porque você não precisará se preocupar com os limites. Você pode executar seus pipelines de dados e, após três execuções, o AWS Glue Data Quality começará a gerar observações e recomendações de regras quando perceber alguma anomalia. Você pode revisar as observações, as estatísticas associadas e incorporar facilmente as recomendações de regras em seu conjunto de regras. Para começar, consulte Como configurar a detecção de anomalias e gerar insights . Observe que os analisadores não afetarão seus índices de qualidade de dados. Eles geram estatísticas que podem ser analisadas ao longo do tempo para gerar observações.

Usando a DetectAnomaly regra

Às vezes, você deseja que seus trabalhos falhem ao detectar anomalias. Para aplicar uma restrição, você deve configurar uma regra. Os analisadores não pararão um trabalho. Em vez disso, eles coletarão estatísticas e analisarão os dados. A configuração da regra `DetectAnomaly` na seção de regras do conjunto de regras confirmará que a verificação da DQ relata que o trabalho não conseguiu aprovar todas as regras da verificação.

Benefícios e casos de uso da detecção de anomalias

Os engenheiros podem gerenciar centenas de pipelines de dados a qualquer momento. Cada pipeline pode extrair dados de diferentes fontes e carregá-los no data lake. Como cada pipeline pode extrair dados de uma fonte diferente e carregá-los no data lake, é difícil obter feedback imediato sobre os dados, independentemente de sua forma ter mudado significativamente ou ter divergido das tendências existentes.

No passado, as fontes de dados upstream mudavam sem aviso prévio às equipes de engenharia de dados, introduzindo hard-to-track “bugs de dados” nesse processo. Adicionar nós do Data Quality às

tarefas facilita muito a vida, pois as tarefas falham quando problemas são detectados. No entanto, isso não elimina todos os modos de falha que preocupam as equipes de dados, o que mantém a porta aberta para a entrada de outros erros de dados.

Um modo de falha está relacionado ao volume de dados. À medida que o armazenamento de dados de uma empresa cresce com o tempo, o número de registros produzidos pelos pipelines de dados pode crescer exponencialmente. Toda semana, as equipes de dados podem precisar atualizar manualmente os trabalhos de ETL para aumentar cada regra do Data Quality que define um limite para o número de linhas ingeridas.

Outro modo de falha é que alguns dos limites das regras de qualidade de dados são muito amplos para acomodar o fato de que o volume de transações varia de acordo com o dia da semana. Nos finais de semana, quase não há transações e, às segundas-feiras, há cerca de três vezes mais transações do que nos outros dias da semana. As equipes de dados têm duas opções: implementar a lógica para alterar o conjunto de regras rapidamente, dependendo do dia, ou definir uma expectativa muito ampla.

Por fim, as equipes de dados também estão preocupadas com erros de dados não muito bem definidos. Os modelos foram treinados com dados com características específicas e, se eles começarem a ficar assimétricos de maneiras inesperadas, a equipe vai querer saber. Por exemplo, em fevereiro, uma empresa pode se expandir para Montana e, assim, as transações que contêm o código “MT” começam a aparecer com mais frequência. Isso pode quebrar a inferência do ML e, como resultado, os modelos previram falsamente que cada transação de Montana era fraudulenta.

É aqui que a detecção de anomalias na qualidade dos dados pode ajudar a resolver esses problemas. Alguns dos benefícios da detecção de anomalias no Data Quality incluem:

- Verificação de dados de forma programada, orientada por eventos ou manual.
- Detecção de anomalias que podem ser indicativas de um evento não intencional, sazonalidade ou uma anormalidade estatística.
- Ofereça recomendações de regras para executar ações em relação às observações encontradas pela detecção de anomalias do Data Quality.

Isso é útil se você deseja:

- detectar anomalias em seus dados automaticamente sem a necessidade de gravar regras de qualidade de dados;

- detectar possíveis problemas em seus dados que as regras de qualidade de dados sozinhas não conseguem encontrar;
- automatizar algumas tarefas que evoluem com o tempo, como limitar o número de linhas ingeridas para o monitoramento da qualidade dos dados.

Permissões do IAM para o AWS Glue Data Quality

Este tópico fornece informações para ajudar você a entender as ações e os recursos que podem ser usados em uma política do AWS Identity and Access Management (IAM) para o AWS Glue Data Quality. Inclui também exemplos de políticas do IAM com as permissões mínimas necessárias para usar o AWS Glue Data Quality com o AWS Glue Data Catalog.

Para obter mais informações sobre grupos de segurança no AWS Glue, consulte Segurança no AWS Glue.

Permissões do IAM para o AWS Glue Data Quality

A tabela a seguir lista as permissões de que um usuário precisa para realizar operações específicas do AWS Glue Data Quality. Para definir uma autorização refinada para o AWS Glue Data Quality, você pode especificar essas ações no elemento `Action` de uma instrução de política do IAM.

Ações do AWS Glue Data Quality

Ação	Descrição	Tipos de recursos
`glue:CreateDataQualityRuleset`	Concede permissão para criar um conjunto de regras de qualidade de dados.	`::dataQualityRuleset/<name>`
`glue:DeleteDataQualityRuleset`	Concede permissão para excluir um conjunto de regras de qualidade de dados.	`::dataQualityRuleset/<name>`
`glue:GetDataQualityRuleset`	Concede permissão para recuperar um conjunto de regras de qualidade de dados.	`::dataQualityRuleset/<name>`
`glue:ListDataQualityRulesets`	Concede permissão para recuperar todos os conjunto	`::dataQualityRuleset/*`

Ação	Descrição	Tipos de recursos
	de regras de qualidade de dados.	
glue:UpdateDataQualityRuleset	Concede permissão para atualizar um conjunto de regras de qualidade de dados.	::dataQualityRuleset/<name>
glue:GetDataQualityResult	Concede permissão para recuperar um resultado de execução de tarefa de qualidade de dados.	::dataQualityRuleset/<name>
glue:ListDataQualityResults	Concede permissão para recuperar todos os resultados de execução de tarefas de qualidade de dados.	::dataQualityRuleset/*
glue:CancelDataQualityRuleRecommendationRun	Concede permissão para interromper uma execução de tarefa de recomendação de qualidade de dados em andamento.	::dataQualityRuleset/*
glue:GetDataQualityRuleRecommendationRun	Concede permissão para recuperar uma execução de tarefa de recomendação de qualidade de dados.	::dataQualityRuleset/*
glue:ListDataQualityRuleRecommendationRuns	Concede permissão para recuperar todas as execuções de tarefas de recomendação de qualidade de dados.	::dataQualityRuleset/*
glue:StartDataQualityRuleRecommendationRun	Concede permissão para iniciar uma execução de tarefa de recomendação de qualidade de dados.	::dataQualityRuleset/*

Ação	Descrição	Tipos de recursos
glue:CancelDataQualityRulesetEvaluationRun	Concede permissão para interromper uma execução de recomendação de tarefa de qualidade de dados em andamento.	::dataQualityRuleset/*
glue:GetDataQualityRulesetEvaluationRun	Concede permissão para recuperar uma execução de uma tarefa de qualidade de dados.	::dataQualityRuleset/*
glue:ListDataQualityRulesetEvaluationRuns	Concede permissão para recuperar todas as execuções de tarefas de qualidade de dados.	::dataQualityRuleset/*
glue:StartDataQualityRulesetEvaluationRun	Concede permissão para iniciar uma execução de tarefa de qualidade de dados.	::dataQualityRuleset/<name>
glue:PublishDataQuality	Concede permissão para publicar resultados de qualidade de dados	::dataQualityRuleset/<name>

A configuração do IAM é requerida para agendar execuções de avaliação

Permissões do IAM

Para que as execuções de avaliação da qualidade de dados agendadas sejam realizadas, você deve adicionar a ação IAM:PassRole à política de permissões.

Permissões do Agendador do AWS EventBridge requeridas

Ação	Descrição	Tipos de recursos
iam:PassRole	Concede permissão para o IAM a fim de permitir que	ARN do perfil usado para chamar StartData

Ação	Descrição	Tipos de recursos
	o usuário passe os perfis aprovados.	`QualityRulesetEvaluationRun`

Sem essas permissões, ocorre o seguinte erro:

```
"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 confiáveis do IAM

Os serviços AWS Glue e Agendados do AWS EventBridge precisam estar listados nas entidades confiáveis para criar e executar uma `StartDataQualityEvaluationRun` agendada.

```
{
    "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"
        }
    ]
}
```

Políticas de exemplo do IAM

Um perfil do IAM para o AWS Glue Data Quality precisa dos seguintes tipos de permissões:

- Permissões para operações do AWS Glue Data Quality para que você possa obter as regras de qualidade de dados recomendadas e executar uma tarefa de qualidade de dados em uma tabela do AWS Glue Data Catalog. Os exemplos de políticas do IAM desta seção incluem as permissões mínimas necessárias para as operações do AWS Glue Data Quality.
- Permissões que concedem acesso à tabela do Data Catalog aos dados subjacentes. Essas permissões variam dependendo do caso de uso. Por exemplo, para dados que você cataloga no Amazon S3, as permissões devem incluir acesso ao Amazon S3.

> **Note**
>
> Você deve configurar as permissões do Amazon S3 além das descritas nesta seção.

Permissões mínimas para obter as regras de qualidade de dados recomendadas

Esse exemplo de política inclui as permissões necessárias para gerar as regras de qualidade de dados 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": "*"
    },
  ]
}
```

Permissões mínimas para executar uma tarefa de qualidade de dados

Esse exemplo de política inclui as permissões necessárias para executar a tarefa de avaliação de qualidade de dados.

As seguintes instruções da política são opcionais dependendo do caso de uso:

- `AllowCloudWatchPutMetricDataToPublishTaskMetrics`: obrigatório se você quiser publicar métricas de execução de qualidade de dados no Amazon CloudWatch.
- `AllowS3PutObjectToWriteTaskResults`: obrigatório se você quiser gravar resultados de execução de qualidade de dados no 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>/*"
    }
  ]
}
```

Introdução ao AWS Glue Data Quality para o Data Catalog

Esta seção de introdução fornece instruções para ajudar você a começar a usar o AWS Glue Data Quality no console da AWS Glue. Você aprenderá a realizar as tarefas essenciais, como gerar recomendações de regras de qualidade de dados e avaliar um conjunto de regras em relação aos seus dados.

Tópicos

Pré-requisitos

Antes de usar o AWS Glue Data Quality, você deve estar familiarizado com o uso do Data Catalog e de crawlers no AWS Glue. Com o AWS Glue Data Quality, você pode avaliar a qualidade das tabelas em um banco de dados do Data Catalog. Você precisará dos seguintes itens:

- Uma tabela do Data Catalog para avaliar segundo o conjunto de regras de qualidade de dados.
- Um perfil do IAM para o AWS Glue Glue, que você fornece quando gera recomendações de regras ou executa uma tarefa de qualidade de dados. Esse perfil deve ter permissão para acessar os recursos que vários processos do AWS Glue Data Quality exigem para realizar a execução em seu nome. Esses recursos incluem o AWS Glue, o Amazon S3 e o CloudWatch. Para ver exemplos de políticas que incluem as permissões mínimas para o AWS Glue Data Quality, consulte Políticas de exemplo do IAM.

  Para saber mais sobre perfis do IAM para o AWS Glue, consulte Create an IAM policy for the AWS Glue service eCreate an IAM role for the AWS Glue service. Você também pode ver uma lista de todas as permissões do AWS Glue que são específicas para qualidade de dados em Authorization for AWS Glue Data Quality actions.
- Um banco de dados com pelo menos uma tabela que contém uma variedade de dados. A tabela usada neste tutorial é denominada `yyz-tickets`, com a tabela `tickets`. Esses dados são uma coleção de informações publicamente disponíveis da cidade de Toronto para multas de estacionamento. Se você criar sua própria tabela, certifique-se de que ela esteja preenchida com uma variedade de dados válidos para obter o melhor conjunto de regras recomendadas.

Exemplo passo a passo

Para ver um exemplo passo a passo com conjuntos de dados de amostra, consulte a postagem do blog sobre o AWS Glue Data Quality.

Gerar recomendações de regras

As recomendações de regras tornam mais fácil começar a trabalhar com qualidade de dados sem precisar escrever código. O AWS Glue Data Quality analisa os dados, identifica as regras e cria um conjunto de regras que você pode avaliar em uma tarefa de qualidade de dados. As execuções de recomendação são excluídas automaticamente após 90 dias.

Para gerar recomendações de regras de qualidade de dados

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, selecione Tables (Tabelas). Em seguida, selecione a tabela para a qual deseja gerar recomendações de regras de qualidade de dados.
3. Na página de detalhes da tabela, selecione a guia Qualidade dos dados para acessar as regras e as configurações do AWS Glue Data Quality para a tabela.

4. Na guia Qualidade dos dados, escolha Adicionar regras e monitorar a qualidade dos dados.
5. Na página Criador de conjuntos de regras, um alerta na parte superior da página solicitará que você inicie uma tarefa de recomendação se não houver nenhuma execução de recomendação de regra.
6. Escolha Recomendar regras para abrir o modal e inserir os parâmetros para a tarefa de recomendação.
7. Escolha um perfil do IAM com acesso ao AWS Glue. Esse perfil deve ter permissão para acessar os recursos que vários processos do AWS Glue Data Quality exigem para realizar a execução em seu nome.
8. Depois que os campos forem preenchidos de acordo com suas preferências, escolha Recomendar regras para iniciar a execução da tarefa de recomendação. Se as execuções de recomendação estiverem em andamento ou tiverem sido concluídas, você poderá gerenciar as execuções neste alerta. Talvez seja necessário atualizar o alerta para ver a alteração de status. As execuções de tarefas de recomendação concluídas e em andamento aparecem na página Histórico de execução, que lista todas as execuções de recomendação dos últimos 90 dias.

O que significam as regras recomendadas

O AWS Glue Data Quality gera regras com base nos dados de cada coluna da tabela de entrada. Ele usa as regras para identificar possíveis limites nos quais os dados podem ser filtrados para manter os requisitos de qualidade. A lista de regras geradas a seguir inclui exemplos que são úteis para entender o que as regras significam e o que elas podem fazer quando aplicadas aos dados.

Para obter uma lista completa dos tipos de regras em Data Quality Definition Language (DQDL) gerados, consulte a referência de tipos de regras de DQDL.

- `IsComplete "SET_FINE_AMOUNT"`: a regra `IsComplete` verifica se a coluna está preenchida para qualquer linha dada. Use essa regra para marcar colunas como não opcionais nos dados.
- `Uniqueness "TICKET_NUMBER" > 0.95`: a `Uniqueness` regra verifica se os dados dentro da coluna atendem a algum limite de exclusividade. Neste exemplo, foi determinado que os dados que preenchem qualquer linha dada para `"TICKET_NUMBER"` têm, no máximo, 95% de conteúdo idêntico a todas as outras linhas, o que sugere essa regra.
- `ColumnValues "PROVINCE" in ["ON", "QC", "AB", "NY",...]`: a regra `ColumnValues` define valores válidos para a coluna, com base no conteúdo existente da coluna. Neste exemplo, os dados de cada linha são uma placa de código de licença de 2 letras para um estado ou província.

- `ColumnLength "INFRACTION_DESCRIPTION" between 15 and 31`: a regra `ColumnLength` impõe uma restrição de tamanho aos dados de uma coluna. Essa regra é gerada a partir dos dados de amostra com base nos comprimentos mínimo e máximo registrados para uma coluna de strings.

Monitorar recomendações de regras

Quando as recomendações de regras de qualidade de dados estão em execução, a página Adicionar regras e monitorar a qualidade de dados exibe informações e ações adicionais que você pode realizar na barra superior.

Quando as recomendações de regras estão em andamento, você pode escolher Parar a execução antes que a tarefa de recomendação seja concluída. Enquanto a tarefa estiver em andamento, você verá o status, em andamento e a data e a hora em que a execução começou.

Quando as recomendações de regras forem concluídas, a barra de recomendação de regras exibirá o número de regras recomendadas, o status da última execução da recomendação e a data e o timestamp em que ela foi concluída.

Você pode adicionar as regras recomendadas escolhendo Inserir recomendação de regra. Para ver as regras recomendadas anteriormente, selecione uma data específica. Para executar uma nova recomendação, escolha Mais ações e depois Regras recomendadas.

Defina as configurações padrão escolhendo Gerenciar configurações do usuário. Você pode definir o caminho padrão para o Amazon S3 armazenar conjuntos de regras ou configurar uma função padrão para executar o catálogo de dados.

Editar conjuntos de regras recomendadas

Como o AWS Glue Data Quality gera regras com base nos dados existentes que você tem disponíveis, é possível ver algumas regras inesperadas ou indesejáveis nas sugestões automatizadas. Para tirar o máximo proveito dos conjuntos de regras recomendadas, você precisa avaliá-los e modificá-los. Nesta etapa do tutorial, você toma as regras geradas na etapa anterior e as ajusta para impor qualidades mais restritivas a alguns dados. Você também relaxa outras regras para garantir que dados corretos e exclusivos possam ser adicionados posteriormente.

Editar um conjunto de regras sugeridas

1. No console do AWS Glue, escolha Catálogo de dados no painel de navegação e depois Adicionar banco de dados. Selecione a tabela `tickets`.
2. Na página de detalhes da tabela, escolha a guia Qualidade dos dados para acessar as regras e as configurações do AWS Glue Data Quality para a tabela.
3. Na seção Conjuntos de regras, selecione o conjunto de regras gerado em Gerar recomendações de regras.
4. Escolha Ações e depois escolha Editar na janela do console. O editor do conjunto de regras é carregado no console. Ele inclui um painel de edição para as regras e uma referência rápida de DQDL.
5. Remova a linha 2 do script. Isso relaxa a exigência de que o tamanho do banco de dados seja restrito a um determinado número de linhas. Após a edição, o arquivo deve conter o seguinte nas linhas de 1 a 3:

```
Rules = [
    IsComplete "TAG_NUMBER_MASKED",
    ColumnLength "TAG_NUMBER_MASKED" between 6 and 9,
```

6. Remova a linha 25 do script. Isso relaxa a exigência de que 96% das províncias registradas sejam ON. Após a edição, o arquivo deve conter o seguinte, da linha 24 até o final do conjunto de regras:

```
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. Altere a linha 14 para o seguinte:

```
IsComplete "TIME_OF_INFRACTION",
```

Isso reforça a exigência na coluna, limitando o banco de dados apenas a multas que contenham a hora da infração registrada. Você deve sempre considerar as multas cuja hora da infração não

está registrada como dados inválidos nesse conjunto de dados. Isso é diferente de situações em que o particionamento ou a transformação podem ser mais apropriados para uso ou inspeção adicional de dados para determinar uma regra de qualidade.

8. Escolha Atualizar conjunto de regras na parte inferior da página do console.

Criar um conjunto de regras

Um conjunto de regras é um grupo de regras de qualidade de dados que você avalia em relação aos seus dados. No console do AWS Glue, você pode criar conjuntos de regras personalizados usando Data Quality Definition Language (DQDL).

Para criar um conjunto de regras de qualidade de dados

1. No console do AWS Glue, escolha Catálogo de dados, Bancos de dados e depois Tabelas no painel de navegação. Selecione a tabela `tickets`.
2. Abra a guia Data quality (Qualidade de dados).
3. Na seção Regras, escolha Criar regra). O editor DQDL é iniciado no console. Ele tem uma área de texto para edição direta e uma referência rápida para as regras do DQDL e o esquema da tabela.
4. Comece a adicionar regras à área de texto do editor DQDL. Você pode escrever as regras diretamente deste tutorial ou usar o atributo de criação de regras Criador de regras DQDL do editor de regras de qualidade de dados.

> **Note**
>
> Como usar o criador de regras DQDL
>
> 1. Selecione um tipo de regra na lista e o sinal de adição para inserir um exemplo de sintaxe no painel do editor.
> 2. Troque os nomes das colunas do espaço reservado por seus próprios nomes de coluna. Os nomes das colunas da tabela estão disponíveis na guia Esquema.
> 3. Atualize o parâmetro de expressão conforme desejar. Para obter uma lista completa de expressões compatíveis com DQDL, consulte Expressões.

Como exemplo, as regras a seguir são restrições para a validação de dados da coluna `ticket_number` na tabela `tickets`. Para adicionar as seguintes regras, use o criador de regras DQDL ou edite diretamente o conjunto de regras:

```
IsComplete "ticket_number",
IsUnique "ticket_number",
ColumnValues "ticket_number" > 9000000000
```

5. Forneça um nome para o novo conjunto de regras no campo Nome do conjunto de regras.
6. Escolha Salvar conjunto de regras.

Avaliar a qualidade dos dados em vários conjuntos de dados

Você pode configurar regras de qualidade de dados em vários conjuntos de dados usando os conjuntos de regras ReferentialIntegrity e DatasetMatch. ReferentialIntegrity verifica se os dados no conjunto de dados primário estão presentes em outros conjuntos de dados.

Para adicionar um conjunto de dados de referência, escolha a guia Esquema e depois Atualizar tabelas de referência. Será solicitado que você selecione um banco de dados e uma tabela. Você pode adicionar a tabela e depois configurar as regras de qualidade dos dados. Tipos de regras como AggregateMatch, RowCountMatch, ReferentialIntegrity, SchemaMatch, and DatasetMatch ofesão compatíveis com a capacidade de executar verificações de qualidade de dados em vários conjuntos de dados.

Como executar um conjunto de regras para avaliar a qualidade de dados

Quando você executa uma tarefa de qualidade de dados, o AWS Glue Data Quality avalia um conjunto de regras em relação aos seus dados e calcula uma pontuação de qualidade de dados. A pontuação representa a porcentagem de regras de qualidade de dados aprovadas para os dados fornecidos.

Para executar uma tarefa de qualidade de dados

1. No console do AWS Glue, escolha Catálogo de dados, Bancos de dados e depois Tabelas no painel de navegação. Selecione a tabela `tickets`.
2. Escolha a guia Qualidade de dados.

3. Na lista Conjuntos de regras, escolha o conjunto de regras segundo as quais você deseja avaliar a tabela. Para esta etapa, recomendamos usar um conjunto de regras que você já tenha escrito ou modificado, em vez de regras geradas. Escolha Executar.
4. No modal, escolha seu perfil do IAM. Esse perfil deve ter permissão para acessar os recursos que vários processos do AWS Glue Data Quality exigem para realizar a execução em seu nome. Você pode salvar o perfil do IAM como padrão ou modificá-lo acessando a página Configuração padrão.
5. Em Data quality actions (Ações de qualidade de dados), escolha se você deseja Publish metrics to Amazon CloudWatch. (Publicar métricas no Amazon CloudWatch). Quando essa opção está selecionada, o AWS Glue Data Quality publica métricas que indicam o número de regras aprovadas e o número de regras reprovadas. Para atuar sobre as métricas armazenadas dessa forma, você pode usar os alarmes do CloudWatch. As principais métricas também são publicadas no Amazon EventBridge para você configurar alertas. Para obter mais informações, consulte Configurar alertas, implantações e agendamentos.
6. Em Frequência de execução, escolha executar sob demanda ou agende o conjunto de regras. Ao agendar um conjunto de regras, é solicitado a você um nome de tarefa. A agenda será criada no Amazon EventBridge. Você pode editar sua agenda no Amazon EventBridge.
7. Para salvar os resultados de qualidade dos dados no Amazon S3, escolha um Local para os resultados de qualidade de dados. O perfil do IAM que você selecionou anteriormente para essa tarefa deve ter acesso de gravação ao local escolhido.
8. Em Configurações adicionais, insira o Número necessário de operadores que você deseja que o AWS Glue aloque para a tarefa de qualidade de dados.
9. Opcionalmente, você pode configurar um filtro na fonte de dados. Isso ajuda a reduzir os dados que você está lendo. Você também pode usar um filtro para executar validações incrementais selecionando as informações da partição e passando-as como parâmetros por meio de chamadas de API. Para melhorar a performance, você pode fornecer um predicado de partição.
10. Escolha Executar. Você deve ver a nova tarefa na lista Data quality task runs (Execuções de tarefas de qualidade de dados). Quando a coluna Status da execução da tarefa for exibida como Concluída, você poderá visualizar os resultados da pontuação de qualidade. Pode ser necessário atualizar a janela do console para que o status seja atualizado corretamente.
11. Para visualizar a coluna dos detalhes do resultado da qualidade dos dados, escolha o ícone “+” para expandir o conjunto de regras. Os resultados mostram as regras em que as regras foram aprovadas e reprovadas na avaliação e o que causou a reprovação da regra.

Como visualizar o índice de qualidade de dados e os resultados

Para ver a última execução em todos os conjuntos de regras criados

1. No console do AWS Glue, escolha Tables (Tabelas) no painel de navegação. Em seguida, selecione a tabela para a qual deseja executar uma tarefa de qualidade de dados.
2. Escolha a guia Qualidade de dados.
3. O resumo da qualidade de dados mostra uma tendência geral das execuções ao longo do tempo. As últimas 10 execuções em todos os conjuntos de regras são exibidas por padrão. Para filtrar por conjunto de regras, selecione o conjunto desejado na lista suspensa. Se houver menos de 10 execuções, todas as execuções concluídas disponíveis serão exibidas.
4. Na tabela Qualidade de dados, cada conjunto de regras com sua última execução (se houver) é mostrado, junto com a pontuação. A expansão do conjunto de regras exibe as regras que estão nesse conjunto de regras, junto com os resultados da regra na execução.

Para ver a última execução de um conjunto de regras específico

1. No console do AWS Glue, escolha Tables (Tabelas) no painel de navegação. Em seguida, selecione a tabela para a qual deseja executar uma tarefa de qualidade de dados.
2. Escolha a guia Qualidade de dados.
3. Na tabela Qualidade de dados, escolha um conjunto de regras específico.
4. Na página de Detalhes do conjunto de regras, escolha a guia Histórico de execuções.

   Todas as execuções de avaliação desse conjunto de regras específico estão listadas na tabela dessa guia. Você pode ver o histórico das pontuações e o status das execuções.
5. Para ver mais informações sobre uma execução específica, escolha a ID da execução para acessar a página Detalhes da execução de avaliação. Nessa página, você pode ver detalhes específicos sobre a execução e mais detalhes sobre o status dos resultados de regras individuais.

Tópicos relacionados

- Referência de tipos de regra DQDL
- Referência de Data Quality Definition Language (DQDL)

Avaliar qualidade de dados com o AWS

O AWS Glue Data Quality avalia e monitora a qualidade dos dados com base em regras que você define. Isso facilita a identificação dos dados que precisam de ação. No AWS Glue Studio, você pode adicionar nós de qualidade de dados ao trabalho visual para criar regras de qualidade de dados em tabelas no catálogo de dados. Você poderá, então, monitorar e avaliar as alterações nos conjuntos de dados à medida que eles evoluírem ao longo do tempo. Para obter uma visão geral de como trabalhar com a qualidade de dados do AWS Glue no AWS Glue Studio, veja o vídeo a seguir.

A seguir estão as etapas de alto nível de como você trabalha com o AWS Glue Data Quality:

1. Criar regras de qualidade de dados: compile um conjunto de regras de qualidade de dados usando o compilador DQDL escolhendo conjuntos de regras integrados que você configura.
2. Configurar um trabalho de qualidade de dados: defina ações com base nos resultados de qualidade de dados e nas opções de saída.
3. Salvar e executar um trabalho com qualidade de dados: criar e executar um trabalho. Salvar o trabalho salvará os conjuntos de regras que você criou para o trabalho.
4. Monitorar e analisar os resultados de qualidade dos dados: analise os resultados da qualidade dos dados após a conclusão da execução do trabalho. Opcionalmente, agende o trabalho para uma data futura.

Benefícios

Analistas de dados, engenheiros de dados e cientistas de dados podem usar o nó Evaluate Data Quality no AWS Glue Studio para analisar, configurar, monitorar e melhorar a qualidade dos dados do editor de trabalho visual. Os benefícios de usar o nó de qualidade de dados incluem:

- Você pode detectar problemas de qualidade de dados: você pode verificar problemas criando regras que verificam as características dos seus conjuntos de dados.
- É fácil começar: você pode começar com regras e ações pré-construídas.
- Integração perfeita: você pode usar nós de qualidade de dados no AWS Glue Studio porque o AWS Glue Data Quality é executado em cima do catálogo de dados do AWS Glue.

Avaliar a qualidade dos dados para trabalhos de ETL no AWS Glue Studio

Neste tutorial, você começa a usar o AWS Glue Data Quality no AWS Glue Studio. Você aprenderá a fazer o seguinte:

- Criar regras usando o compilador de regras da Data Quality Definition Language (DQDL).
- Especificar ações de qualidade de dados, dados a serem produzidos e o local de saída dos resultados de qualidade de dados.
- Analisar os resultados de qualidade dos dados.

Para praticar com um exemplo, revise a postagem do blog Getting started with AWS Glue Data Quality for ETL pipelines.

Etapa 1: adicionar o nó de transformação Evaluate Data Quality à tarefa visual

Nesta etapa, você adiciona o nó Evaluate Data Quality à tarefa visual

Para adicionar o nó de qualidade de dados

1. No console Glue Studio do AWS, escolha Visual com origem e destino na seção Criar tarefa e, em seguida, escolha Criar.
2. Escolha um nó ao qual você deseja aplicar a transformação de qualidade de dados. Normalmente, isso será um nó de transformação ou uma fonte de dados.
3. Abra o painel de recursos à esquerda escolhendo o ícone “+”. Em seguida, procure Evaluate Data Quality na barra de pesquisa e escolha Evaluate Data Quality nos resultados da pesquisa.
4. O editor visual de tarefas mostrará a ramificação do nó de transformação Evaluate Data Quality a partir do nó selecionado. No lado direito do console, a guia Transform (Transformar) é aberta automaticamente. Se você precisar alterar o nó principal, escolha a guia Propriedades do nó e, em seguida, escolha o nó principal no menu suspenso.

   Quando você escolhe um novo nó principal, uma nova conexão é feita entre o nó superior e o nó Evaluate Data Quality (Avaliar qualidade dos dados). Remova todos os nós principais indesejados. Somente um nó principal pode ser conectado a um nó Evaluate Data Quality.
5. A transformação Evaluate Data Quality oferece suporte a vários pais para que você possa validar as regras de qualidade de dados em vários conjuntos de dados. As regras que oferecem suporte a vários conjuntos de dados incluem ReferentialIntegrity, DatasetMatch, SchemaMatch, RowCountMatch e AggregateMatch.

Ao adicionar várias entradas à transformação Evaluate Data Quality, você precisa selecionar sua entrada “primária”. Sua entrada principal é o conjunto de dados para o qual você deseja validar a qualidade dos dados. Todos os outros nós ou entradas são tratados como referências.

Você pode usar a transformação Evaluate Data Quality para identificar registros específicos que falharam nas verificações de qualidade dos dados. Recomendamos que você escolha seu conjunto de dados principal porque novas colunas que sinalizam registros incorretos são adicionadas ao conjunto de dados primário.

6. Você pode especificar aliases para fontes de dados de entrada. Os aliases fornecem outra forma de referenciar a fonte da entrada quando você está usando a regra ReferentialIntegrity. Como somente uma fonte de dados pode ser designada como fonte primária, cada fonte de dados adicional adicionada exigirá um alias.

   No exemplo a seguir, a regra ReferentialIntegrity especifica a fonte de dados de entrada pelo nome do alias e realiza uma comparação individual com a fonte de dados primária.

```
Rules = [
 ReferentialIntegrity “Aliasname.name” = 1
]
```

Etapa 2: criar uma regra usando DQDL

Nesta etapa, você cria uma regra usando DQDL. Para este tutorial, você criará uma única regra usando o tipo de regra Completeness. Esse tipo de regra verifica a porcentagem de valores completos (não nulos) em uma coluna em relação a uma determinada expressão. Para obter mais informações sobre o uso de DQDL, consulte DQDL.

1. Na guia Transformar, adicione um tipo de regra clicando no botão Inserir. Isso adiciona o tipo de regra ao editor de regras, onde você pode inserir os parâmetros da regra.

   > **Note**
   >
   > Ao editar regras, verifique se as regras estão entre colchetes e certifique-se de que as regras estejam separadas por vírgulas. Por exemplo, uma expressão de regra completa terá a seguinte aparência:

```
Rules= [
    Completeness "year">0.8, Completeness "month">0.8
]
```

Este exemplo especifica o parâmetro de completude para as colunas denominadas 'ano' e 'mês'. Para que a regra seja aprovada, essas colunas devem estar mais de 80% 'completas' ou ter dados em mais de 80% das instâncias para cada coluna respectiva.

Neste exemplo, pesquise e insira o tipo de regra Completeness (Completude). Isso adiciona o tipo de regra ao editor de regras. Esse tipo de regra tem a seguinte sintaxe: `Completeness <COL_NAME> <EXPRESSION>`.

A maioria dos tipos de regras exige que você forneça uma expressão como parâmetro para criar uma resposta booleana. Para obter mais informações sobre expressões DQDL compatíveis, consulte Expressões DQDL. Em seguida, você adicionará o nome da coluna.

2. No compilador de regras DQDL, clique na guia Esquema. Use a barra de pesquisa para localizar o nome da coluna no esquema de entrada. O esquema de entrada exibe o nome da coluna e o tipo de dados.
3. No editor de regras, clique à direita do tipo de regra para inserir o cursor onde a coluna será inserida. Como alternativa, você pode digitar o nome da coluna na regra.

   Por exemplo, na lista de colunas na lista de esquemas de entrada, clique no botão Inserir ao lado da coluna (neste exemplo, ano). Isso adicionará a coluna à regra.
4. Em seguida, no editor de regras, adicione uma expressão para avaliar a regra. Como o tipo de regra Completeness verifica a porcentagem de valores completos (não nulos) em uma coluna em relação a uma determinada expressão, insira uma expressão como `> 0.8`. Essa regra verificará se a coluna tem mais de 80% de valores completos (não nulos).

Etapa 3: configurar saídas de qualidade de dados

Depois de criar regras de qualidade de dados, você pode selecionar opções adicionais para especificar a saída do nó de qualidade de dados.

1. Em Data quality transform output (Saída de transformação de qualidade de dados), escolha uma das seguintes opções:

   - Original data: escolha para saída dos dados de entrada originais. Quando você escolhe essa opção, um novo nó filho “rowLevelOutcomes” é adicionado ao trabalho. O esquema corresponde ao esquema do conjunto de dados primário que foi passado como entrada para a transformação. Essa opção é útil se você quiser apenas transmitir os dados e rejeitar o trabalho quando ocorrerem problemas de qualidade.

     Outro caso de uso é quando você deseja detectar registros incorretos que falharam nas verificações de qualidade dos dados. Para detectar registros incorretos, escolha a opção Adicionar novas colunas para indicar erros na qualidade dos dados. Essa ação adiciona quatro novas colunas ao esquema da transformação “rowLevelOutcomes”.

     - DataQualityRulesPass (matriz de strings): Fornece uma matriz de regras que passaram pelas verificações de qualidade dos dados.
     - DataQualityRulesFail (matriz de strings): Fornece uma matriz de regras que foram reprovadas pelas verificações de qualidade dos dados.
     - DataQualityRulesSkip (matriz de strings): Fornece uma matriz de regras que foram ignoradas. As regras a seguir não podem identificar registros de erro porque são aplicadas no nível do conjunto de dados.
       - AggregateMatch
       - ColumnCount
       - ColumnExists
       - ColumnNamesMatchPattern
       - CustomSql
       - RowCount
       - RowCountMatch
       - StandardDeviation
       - Média
       - ColumnCorrelation
     - DataQualityEvaluationResult: fornece o status “Aprovado” ou “Falha” no nível da linha. Observe que seu resultado geral pode ser FALHA, mas um determinado registro pode ser aprovado. Por exemplo, a regra RowCount pode ter falhado, mas todas as outras regras podem ter sido bem-sucedidas. Nesses casos, o status desse campo é “Aprovado”.

2. Resultados de qualidade dos dados: opte pela saída das regras configuradas e seu status de aprovação ou reprovação. Essa opção é útil se você quiser gravar seus resultados no Amazon S3 ou em outros bancos de dados.
3. Configurações de saída de qualidade de dados (Opcional): escolha Configurações de saída de qualidade de dados para revelar o campo Local do resultado da qualidade de dados. Em seguida, clique em Procurar para procurar um local do Amazon S3 para definir como objetivo de saída de qualidade de dados.

Etapa 4. Configurar ações de qualidade de dados

Você pode usar ações para publicar métricas do CloudWatch ou para interromper trabalhos com base em critérios específicos. As ações só estarão disponíveis depois que você criar uma regra. Quando você escolhe essa opção, as mesmas métricas também são publicadas no Amazon EventBridge. Você pode usar essas opções para criar alertas para notificação.

- Em caso de falha no conjunto de regras: você pode escolher o que fazer se um conjunto de regras falhar durante a execução do trabalho. Se você quiser que o trabalho falhe se a qualidade dos dados falhar, escolha quando o trabalho deve falhar selecionando uma das opções a seguir. Por padrão, essa ação não é selecionada e a tarefa concluirá sua execução mesmo se as regras de qualidade de dados falharem.
  - Nenhum: se você escolher Nenhum (padrão), o trabalho não falhará e continuará sendo executado apesar das falhas no conjunto de regras.
  - Falha na tarefa após carregar os dados no destino: a tarefa falha e nenhum dado é salvo. Para salvar os resultados, escolha um local do Amazon S3 onde os resultados de qualidade dos dados serão salvos.
  - Falha na tarefa sem carregar os dados de destino: essa opção causa falha na tarefa imediatamente quando ocorre um erro de qualidade de dados. Ela não carrega nenhum destino de dados, incluindo os resultados da transformação da qualidade dos dados.

Etapa 5: visualizar os resultados de qualidade dos dados

Depois de executar o trabalho, visualize os resultados de qualidade dos dados clicando na guia Qualidade dos dados.

1. Para cada trabalho executado, veja os resultados de qualidade dos dados. Cada nó exibe um status de qualidade de dados e detalhes de status. Clique em um nó para ver todas as regras e o status de cada regra.
2. Escolha Baixar resultados para baixar um arquivo CSV que contém informações sobre a execução do trabalho e os resultados da qualidade dos dados.
3. Se você tiver mais de uma execução de trabalho com resultados de qualidade de dados, poderá filtrar os resultados por intervalo de data e hora. Clique em Filtrar por um intervalo de data e hora para expandir a janela do filtro.
4. É possível escolher entre intervalo relativo e intervalo absoluto. Para intervalos absolutos, use o calendário para selecionar uma data e insira valores para hora de início e hora de término. Após terminar, escolha Aplicar.

Compilador de regras de qualidade de dados

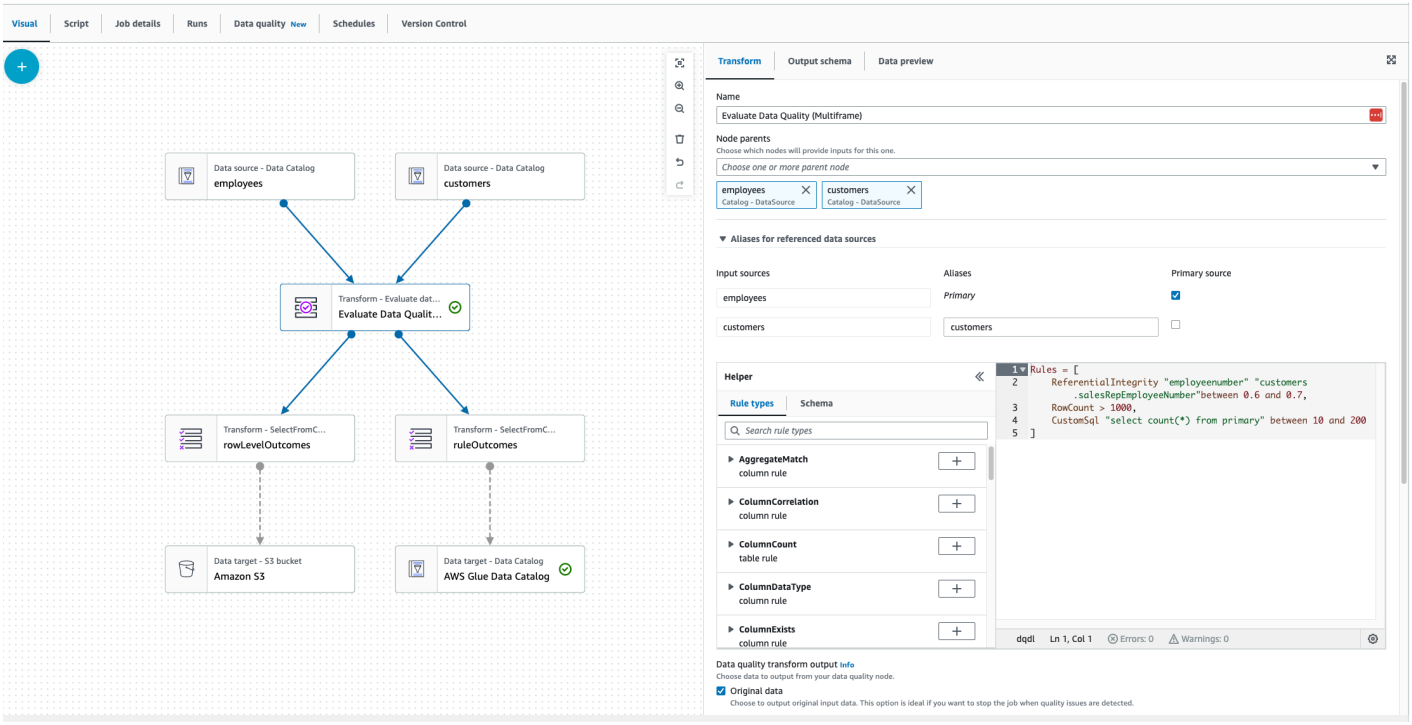
Com o criador de regras da Data Quality Definition Language (DQDL), você pode criar regras de qualidade de dados para avaliar seus dados. Comece selecionando um tipo de regra e depois especifique os parâmetros no editor de regras. O editor de regras também mostra erros e avisos à medida que você criar regras.

O guia do DQDL fornece documentação abrangente sobre como estruturar regras usando a sintaxe, os tipos e os exemplos de regras integradas do DQDL.

Nó Evaluate Data Quality

Ao trabalhar com o nó de transformação Evaluate Data Quality e o compilador de regras DQDL, você pode expandir o espaço de trabalho.

- Para expandir a guia Transformar para preencher a tela inteira, escolha o ícone de expansão no canto superior direito do painel de detalhes do nó.
- Para expandir o editor de regras DQDL, escolha o ícone << para expandir o editor de regras e fechar as guias Tipos de regras e Esquema.



Componentes

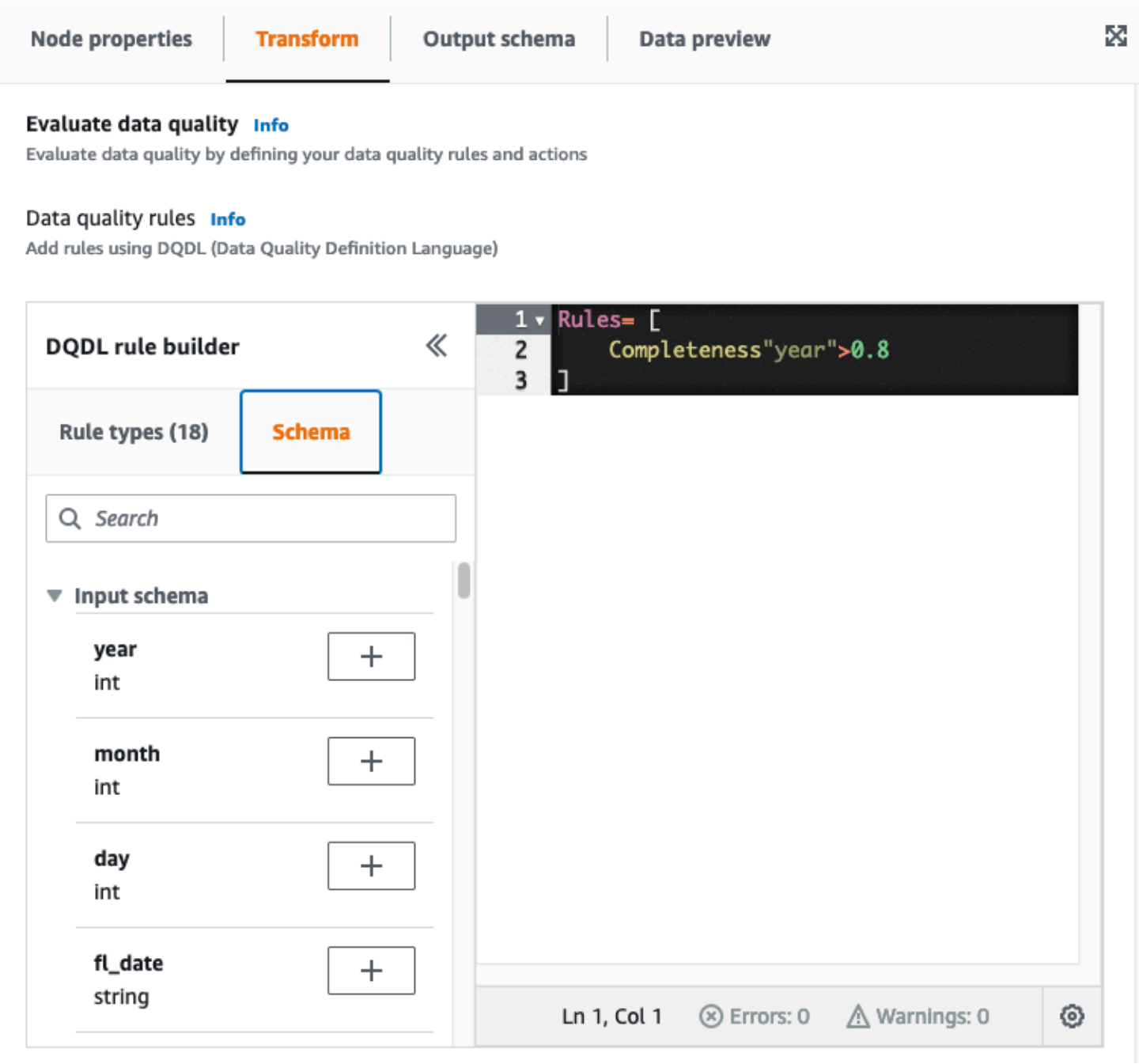
Existem 26 tipos de regras que são incorporados ao AWS Glue Studio. Cada tipo de regra tem uma descrição e exemplos de como elas podem ser usadas.

Tipos de regras de qualidade de dados

O AWS Glue Studio fornece tipos de regras integrados para facilitar a criação de uma regra. Para obter mais informações sobre tipos de regras, consulte Referência de tipos de regras DQDL.

Esquema

A guia Schema (Esquema) exibe os nomes das colunas e o tipo de dados do nó principal. Esquemas de vários nós são exibidos. Você pode visualizar o esquema de entrada, pesquisar pelo nome da coluna e inserir a coluna no editor de regras.



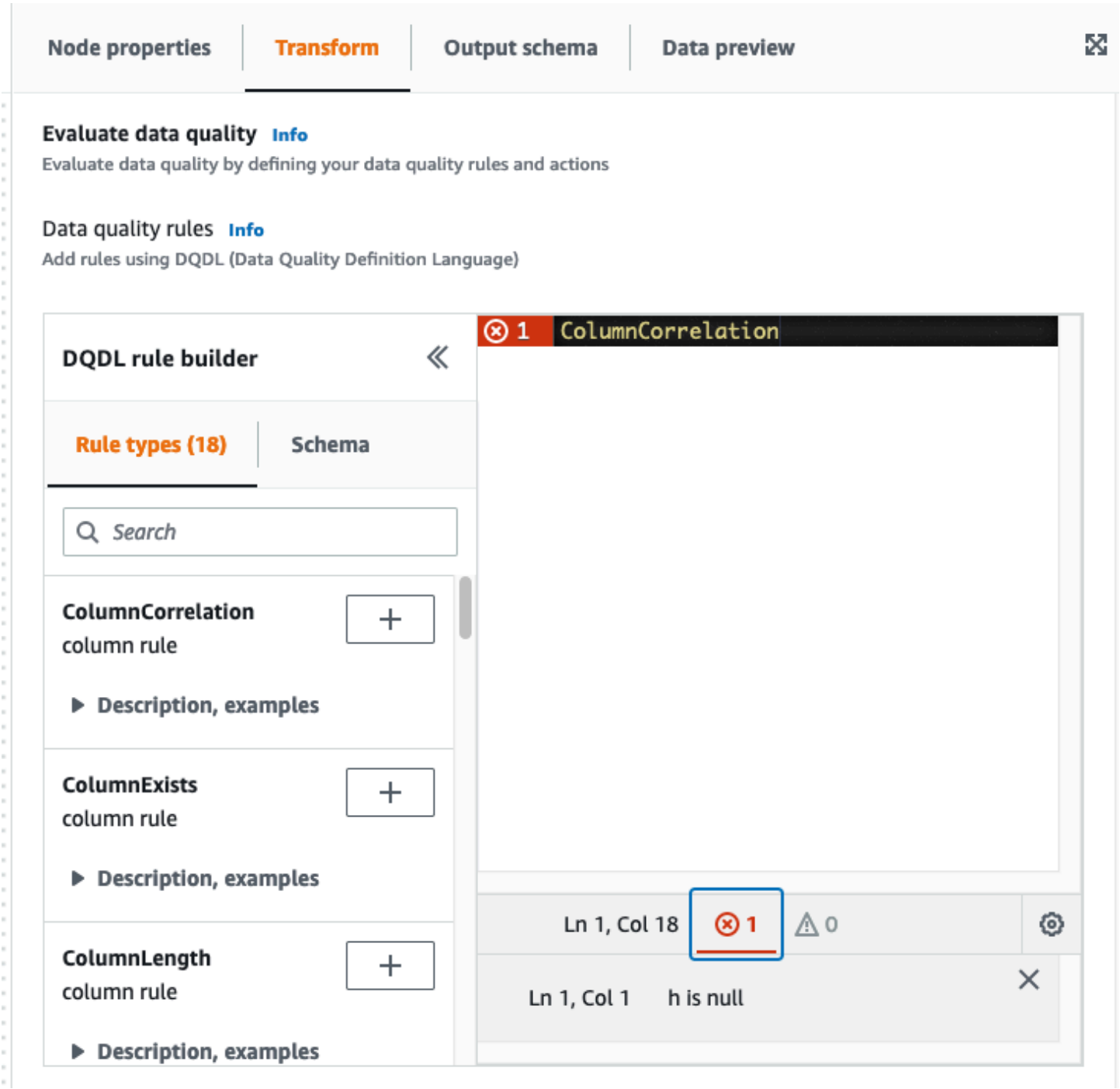
Editor de regras

O editor de regras é um editor de texto em que você pode escrever e editar regras. Se você selecionar um tipo de regra no compilador de regras DQDL, o tipo de regra será adicionado ao editor de regras. Em seguida, você pode especificar parâmetros, adicionar regras e editar regras conforme necessário, modificando o texto. O AWS Glue Studio valida as regras no editor de regras e exibe erros e avisos, se houver.

Erros e advertências

Se uma regra não seguir a sintaxe da regra DQDL, o editor de regras mostra vários indicadores visuais de que há um erro:

- O editor de regras exibe um ícone de erro e a linha com o erro em vermelho.
- O editor de regras exibe o número de erros ao lado do ícone vermelho de erro.
- Quando você escolhe a linha com o erro, uma descrição e o local (linha e coluna) do erro são exibidos na parte inferior do editor de regras.



Ações de qualidade de dados

Por padrão, essa ação não é selecionada e o trabalho concluirá sua execução mesmo se as regras de qualidade de dados falharem.

Escolha entre as ações a seguir. Você pode usar ações para publicar resultados no CloudWatch ou interromper trabalhos com base em critérios específicos. As ações só estarão disponíveis depois que você criar uma regra.

- Publicar resultados no CloudWatch: ao executar um trabalho, adicione os resultados ao CloudWatch.
- Reprovar o trabalho quando a qualidade dos dados for reprovada: se as regras de qualidade de dados forem reprovadas, o trabalho também será reprovado como resultado.

Saída da transformação Data quality

- Dados originais: escolha a saída dos dados de entrada originais. Essa opção é ideal se você quiser interromper o trabalho quando problemas de qualidade forem detectados.
- Métricas de qualidade de dados: opte pela saída das regras configuradas e o status de aprovação ou reprovação correspondente. Essa opção é útil se você quiser fazer uma ação personalizada.

Configurações de saída de qualidade de dados

Defina a localização do resultado de qualidade de dados especificando o local do Amazon S3 como o destino de saída de qualidade de dados.

Como configurar a detecção de anomalias e gerar insights

O AWS Glue Data Quality (DQ) avalia seus dados com base nas regras de qualidade de dados que você grava e fornece insights e observações sobre seus dados ao longo do tempo para que você possa executar ações imediatas. Como o DQ verifica seus dados, ele calcula métricas estatísticas, como contagem de linhas, máximo ou mínimo, e as compara com expressões de limite.

Alguns dos benefícios da detecção de anomalias do Data Quality incluem:

- verificação automática contínua de dados;
- detecção de anomalias que podem ser indicativas de um evento não intencional ou anormalidade estatística;
- fornecimento de recomendações de regras para agir com relação às observações encontradas pela detecção de anomalias no Data Quality.

Isso é útil se você deseja:

- detectar automaticamente anomalias em seus dados, sem a necessidade de gravar uma qualidade de dados;
- criar um perfil de dados e visualizar representações visuais da aparência dos dados;
- acompanhar como seus dados mudam ao longo do tempo.

Quais observações sobre meus dados posso ver?

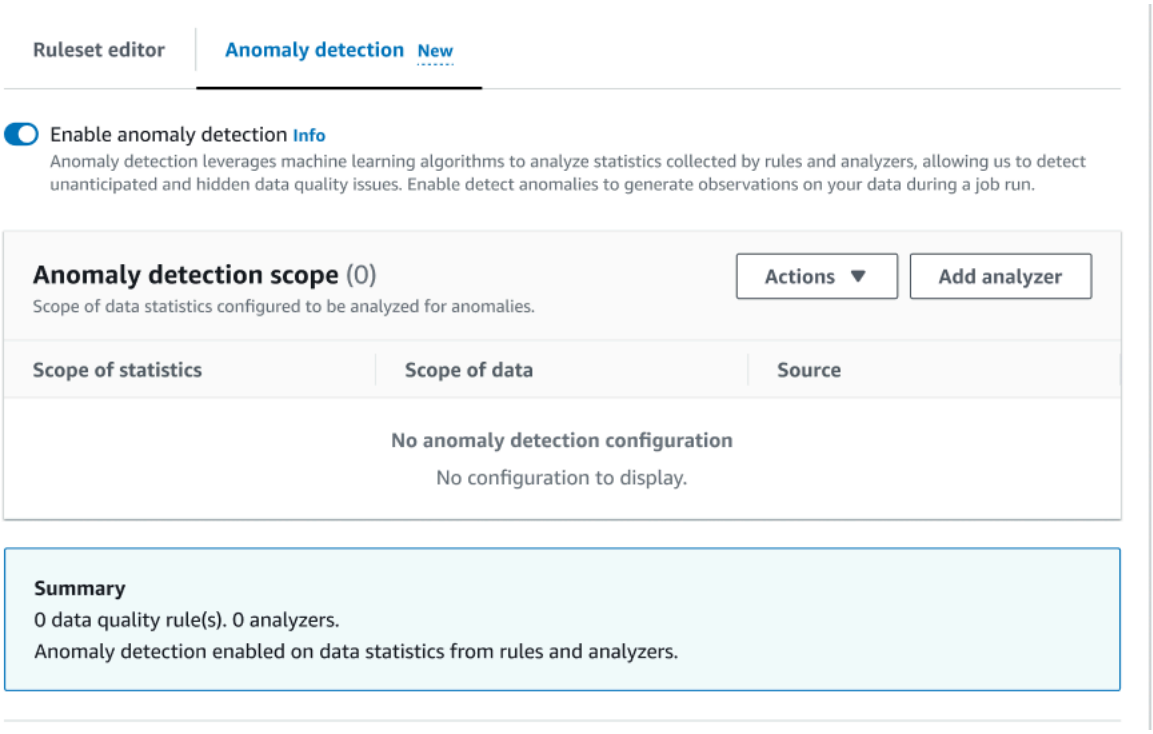
O DQ identifica valores discrepantes nas estatísticas de dados coletadas, mudanças nos formatos de dados, desvios de dados e mudanças no esquema. Com base em observações, o DQ recomenda regras de qualidade de dados que os usuários possam operacionalizar facilmente. As estatísticas incluem integridade, exclusividade, média, soma StandardDeviation, entropia e. DistinctValuesCount UniqueValueRatio

Como habilitar a detecção de anomalias no AWS Glue Studio

Para habilitar a detecção de anomalias, você pode abrir um trabalho do AWS Glue Studio e ativar a opção “Habilitar detecção de anomalias”. Ativar essa opção permite a detecção de anomalias em seus dados, analisando-os ao longo do tempo e fornecendo estatísticas de dados sobre seus dados e observações com base nas quais você pode agir.

Para habilitar a detecção de anomalias no AWS Glue Studio:

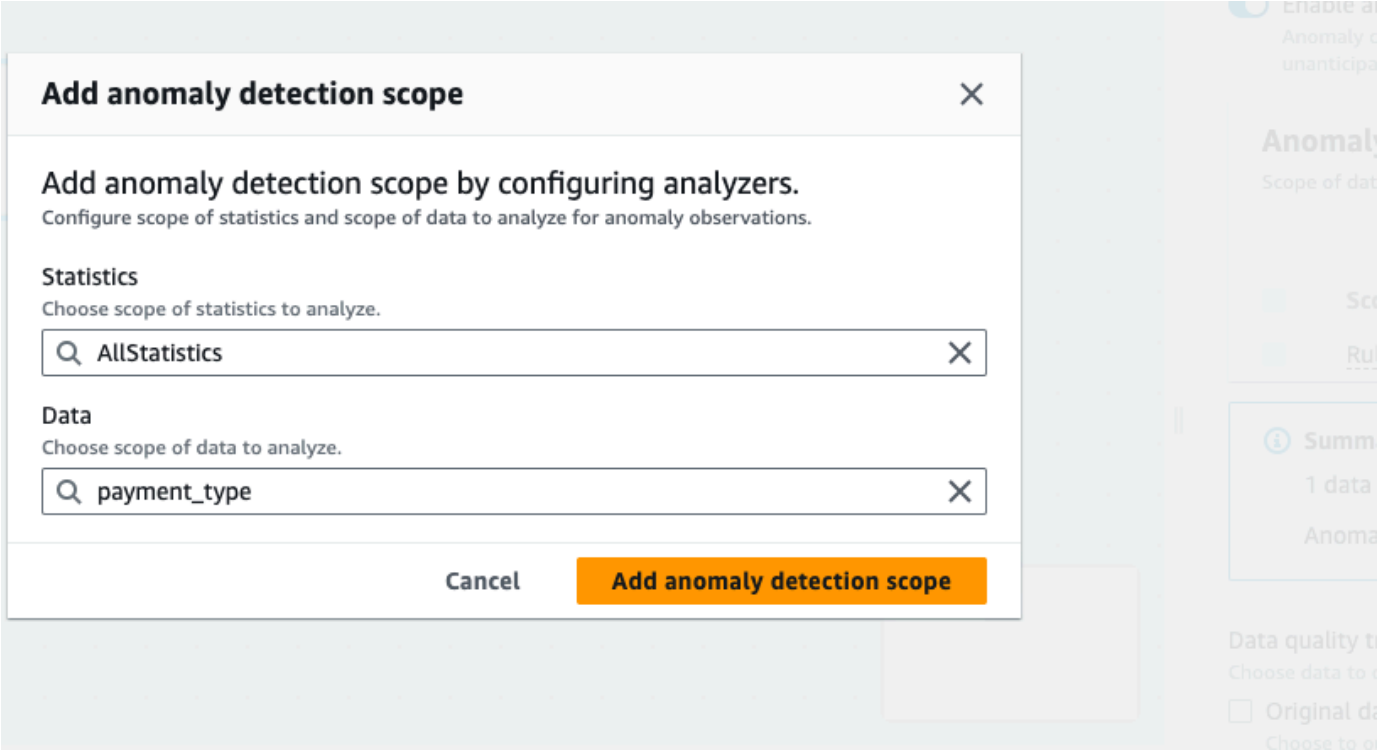
1. Escolha o nó do Data Quality em seu trabalho e, em seguida, escolha a guia Detecção de anomalias. Ative a opção “Habilitar detecção de anomalias”.



2. Defina os dados para monitorar anomalias ao escolher Adicionar analisador. Há dois campos que você pode preencher: Estatísticas e Dados.

   O campo “Estatísticas” refere-se a informações sobre a forma e outras propriedades dos dados. É possível escolher uma ou mais estatísticas por vez ou escolher todas as estatísticas. As estatísticas incluem: integridade, exclusividade, média, soma StandardDeviation, entropia e. DistinctValuesCount UniqueValueRatio

   O campo “Dados” refere-se às colunas no seu conjunto de dados. Você pode escolher todas as colunas ou colunas individuais.



3. Escolha Adicionar escopo de detecção de anomalias para salvar as alterações. Depois de criar analisadores, você pode vê-los na seção Escopo de detecção de anomalias.

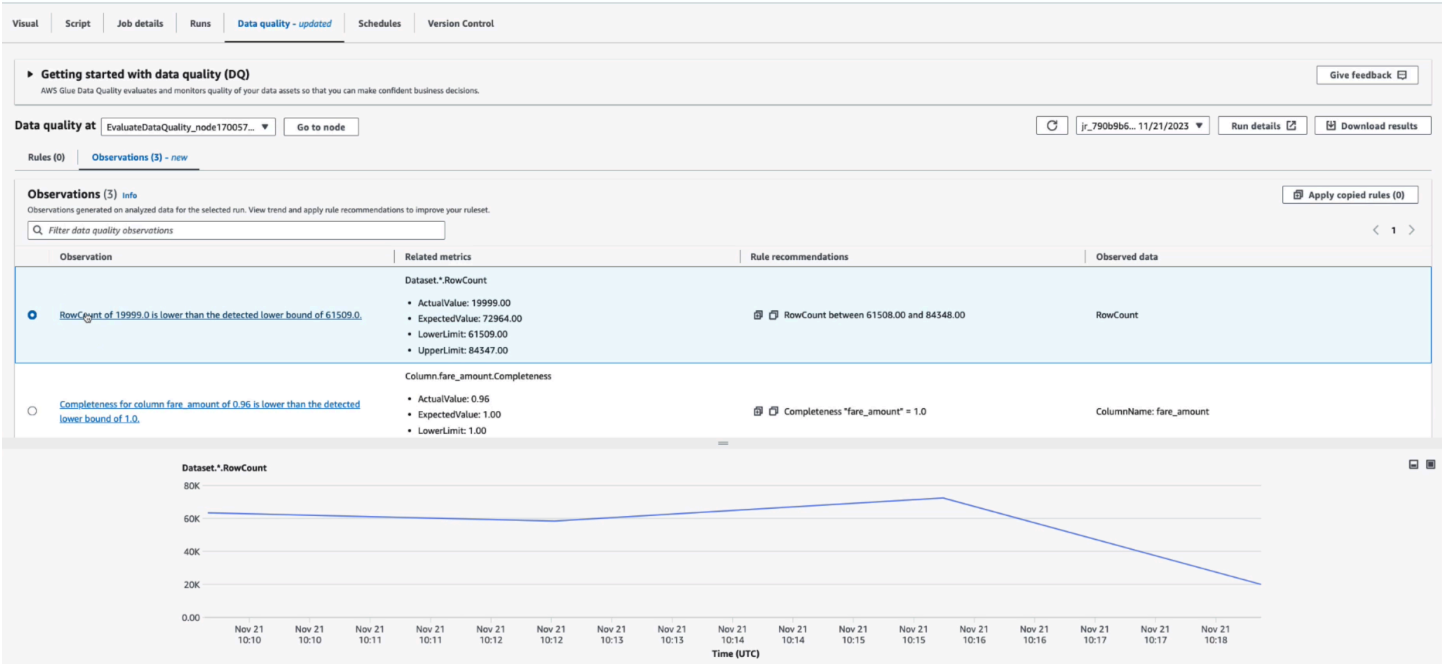
   Você também pode usar o menu Ações para editar seus analisadores ou escolher a guia Editor de conjunto de regras e editar o analisador diretamente no bloco de notas do editor de conjunto de regras. Você verá os analisadores salvos logo abaixo de todas as regras criadas.

```
Rules = [
]

Analyzers = [
    Completeness “id”
]
```

   Com o conjunto de regras atualizado junto com os analisadores, o Data Quality monitora continuamente os dados recebidos, sinalizando anomalias por meio de alertas ou paradas de trabalho com base em suas configurações.

> **Note**
>
> As observações são geradas quando um mínimo de três valores por estatística de dados são observados em seu conjunto de dados. Se não houver observações visíveis, o Data Quality não tem dados suficientes para gerar uma observação. Depois de vários trabalhos executados, o Data Quality pode fornecer informações sobre seus dados e os exibirá na seção Observações.

Os analisadores geram observações detectando anomalias em seus dados e fornecem recomendações para criar regras progressivamente. É possível visualizar as observações ao escolher a guia Data Quality. As observações são específicas para cada execução de trabalho. Você pode visualizar o nó específico do Data Quality e a execução do trabalho na parte superior da seção Observações. Escolha um novo nó ou execução de trabalho para visualizar observações específicas desse nó e trabalho.



Observação: cada insight é baseado em uma execução de trabalho específica configurada pelos conjuntos de regras e analisadores que você especificou.

Métricas relacionadas: quando as observações são geradas, a coluna “Métricas relacionadas” mostra a regra e os valores reais e esperados, bem como os limites inferior e superior.

Recomendações de regras: em seguida, o AWS Glue também recomenda regras para resolver essa questão. Cada regra recomendada pode ser copiada clicando no ícone de cópia. Você pode copiar todas as regras recomendadas clicando no ícone de cópia ao lado de cada regra e clicando em Aplicar regras copiadas.

Dados monitorados: a coluna “Dados monitorados” fornece a coluna ou a linha que foi monitorada e acionou a observação.

Como aplicar uma regra recomendada ao seu nó do Data Quality

Depois que uma observação for gerada e uma regra recomendada for fornecida, você poderá aplicar essa regra ao seu nó de qualidade de dados. Para fazer isso:

1. Clique no ícone de cópia ao lado de cada recomendação de regra. Isso adicionará a recomendação da regra a um bloco de notas que você poderá recuperar posteriormente.
2. Clique em Aplicar recomendações de regras. Isso abre o bloco de notas onde você pode ver as regras que você copiou anteriormente.
3. Escolha Copiar regras.
4. Escolha Aplicar ao editor de conjunto de regras. Essa ação abre o editor de conjunto de regras em que você pode colar as regras copiadas.
5. Cole as regras copiadas no editor do conjunto de regras.

Qualidade de dados para trabalhos de ETL em notebooks AWS Glue Studio

Neste tutorial, você aprende a usar o AWS Glue Data Quality para extração, transformação e carregamento (ETL) em cadernos do AWS Glue Studio.

Você pode usar cadernos no AWS Glue Studio para editar scripts de trabalhos e visualizar a saída sem precisar executar um trabalho inteiro. Você também pode adicionar markdown e salvar cadernos como arquivos do `.ipynb` e scripts de trabalho. Observe que é possível iniciar um caderno sem instalar software localmente nem gerenciar servidores. Quando estiver satisfeito com seu código, você pode usar o AWS Glue Studio para converter facilmente seu caderno em um trabalho do AWS Glue.

O conjunto de dados que você usa neste exemplo consiste nos dados de pagamento de um provedor da Medicare obtidos baixando dois conjuntos de dados do Data.CMS.gov: "Inpatient

Prospective Payment System Provider Summary for the Top 100 Diagnosis-Related Groups - FY2011" e "Inpatient Charge Data FY 2011".

Depois de fazer download dos dados, nós os modificamos para apresentar alguns registros errados ao final do arquivo. Esse arquivo modificado está localizado em um bucket público do Amazon S3 em `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv`.

Pré-requisitos

- Função do AWS Glue com a permissão do Amazon S3 para gravar no seu bucket Amazon S3 de destino
- Um novo caderno (consulte Getting started with notebooks in AWS Glue Studio)

Criar um trabalho de ETL no AWS Glue Studio

Para criar um trabalho de ETL

1. Altere a versão da sessão para AWS Glue 3.0.

   Para fazer isso, remova todas as células de código padronizado com a seguinte mágica e execute a célula. Observe que esse código padronizado é fornecido automaticamente na primeira célula quando um novo caderno é criado.

   ```
   %glue_version 3.0
   ```

2. Copie código a seguir no e execute na célula.

   ```
   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. Na próxima célula, importe a classe `EvaluateDataQuality` que avalia AWS Glue Data Quality.

```
from awsgluedq.transforms import EvaluateDataQuality
```

4. Na próxima célula, leia os dados de origem usando o arquivo .csv que está armazenado no bucket público do 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. Converta os dados em um AWS Glue DynamicFrame.

```
from awsglue.dynamicframe import DynamicFrame
medicare_dyf = DynamicFrame.fromDF(medicare,glueContext,"medicare_dyf")
```

6. Crie o conjunto de regras usando a Data Quality Definition Language (DQDL).

```
EvaluateDataQuality_ruleset = """
    Rules = [
        ColumnExists "Provider Id",
        IsComplete "Provider Id",
        ColumnValues  " Total Discharges " > 15
]
    ]
"""
```

7. Valide o conjunto de dados em relação ao conjunto de regras.

```
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. Reveja os resultados.

```
ruleOutcomes = SelectFromCollection.apply(
    dfc=EvaluateDataQualityMultiframe,
    key="ruleOutcomes",
    transformation_ctx="ruleOutcomes",
)

ruleOutcomes.toDF().show(truncate=False)
```

Saída:

```
-------------------------------------+-------
+-----------------------------------------------------
+-----------------------------------------+
|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 as linhas aprovadas e revise as linhas com falha nos resultados em nível de linha do Data Quality.

```
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
```

Saída:

```
+----------------------------------------+-----------
+-------------------------------------+-------------------------+-------------
+--------------+-----------------+-----------------------------------
+------------------+-------------------------+----------------------
+-------------------------+-------------------------
+-----------------------------------------+---------------------------
+---------------------------+
|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
```

Observe que o AWS Glue Data Quality adicionou quatro novas colunas (DataQualityRulesPass DataQualityRulesFail DataQualityRulesSkip,, e DataQualityEvaluationResult). Isso indica os registros aprovados, os registros que falharam, as regras ignoradas para a avaliação em nível de linha e os resultados gerais em nível de linha.

10. Grave a saída em um bucket do Amazon S3 para analisar os dados e visualizar os 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")
```

Referência de Data Quality Definition Language (DQDL)

A Data Quality Definition Language (DQDL) é uma linguagem específica de domínio que você usa para definir regras para o AWS Glue Data Quality.

Este guia apresenta os principais conceitos de DQDL para ajudar você a entender a linguagem. Ele também fornece uma referência para tipos de regras DQDL com sintaxe e exemplos. Antes de usar este guia, recomendamos que você se familiarize com o AWS Glue Data Quality. Para ter mais informações, consulte AWS Glue Qualidade de dados.

> **Note**
>
> DynamicRules só são compatíveis com AWS Glue ETL.

Sumário

Sintaxe DQDL

Um documento DQDL diferencia maiúsculas de minúsculas e contém um conjunto de regras que agrupa regras individuais de qualidade de dados. Para estruturar um conjunto de regras, você deve criar uma lista denominada `Rules` (em maiúsculas), delimitada por colchetes. A lista deve conter uma ou mais regras DQDL separadas por vírgula, como no exemplo a seguir.

```
Rules = [
   IsComplete "order-id",
   IsUnique "order-id"
]
```

Estrutura da regra

A estrutura de uma regra DQDL depende do tipo de regra. No entanto, as regras DQDL geralmente se encaixam no formato a seguir.

```
<RuleType> <Parameter> <Parameter> <Expression>
```

`RuleType` é o nome com distinção entre maiúsculas e minúsculas do tipo de regra que você quer configurar. Por exemplo, `IsComplete`, `IsUnique` ou `CustomSql`. Os parâmetros da regra diferem para cada tipo de regra. Para obter uma referência completa dos tipos de regras DQDL e seus parâmetros, consulte Referência de tipos de regra DQDL.

Regras compostas

A DQDL é compatível com os seguintes operadores lógicos que você pode usar para combinar regras. Essas regras são chamadas de regras compostas.

e

O operador lógico `and` resultará em `true` se, e somente se, as regras que ele conecta forem `true`. Caso contrário, a regra combinada resultará em `false`. Cada regra que você conecta com o operador `and` deve estar entre parênteses.

O exemplo a seguir usa o operador `and` para combinar duas regras DQDL.

```
(IsComplete "id") and (IsUnique "id")
```

or

O operador lógico `or` resultará em `true` se, e somente se, uma ou mais regras que ele conecta forem `true`. Cada regra que você conecta com o operador `or` deve estar entre parênteses.

O exemplo a seguir usa o operador `or` para combinar duas regras DQDL.

```
(RowCount "id" > 100) or (IsPrimaryKey "id")
```

Você pode usar o mesmo operador para conectar várias regras, portanto, a combinação de regras a seguir é permitida.

```
(Mean "Star_Rating" > 3) and (Mean "Order_Total" > 500) and (IsComplete "Order_Id")
```

Porém, você não pode combinar os operadores lógicos em uma única expressão. Por exemplo, a operação a seguir não é permitida.

```
(Mean "Star_Rating" > 3) and (Mean "Order_Total" > 500) or (IsComplete "Order_Id")
```

Como as regras compostas funcionam

Por padrão, as regras compostas são avaliadas como regras individuais em todo o conjunto de dados ou tabela e, em seguida, os resultados são combinados. Em outras palavras, a coluna inteira é avaliada primeiro e, em seguida, o operador é aplicado. Esse comportamento padrão é explicado abaixo com um exemplo:

```
# Dataset

+------+------+
|myCol1|myCol2|
+------+------+
|     2|     1|
|     0|     3|
+------+------+

# Overall outcome

+-----------------------------------------------------------+-------+
|Rule                                                       |Outcome|
```

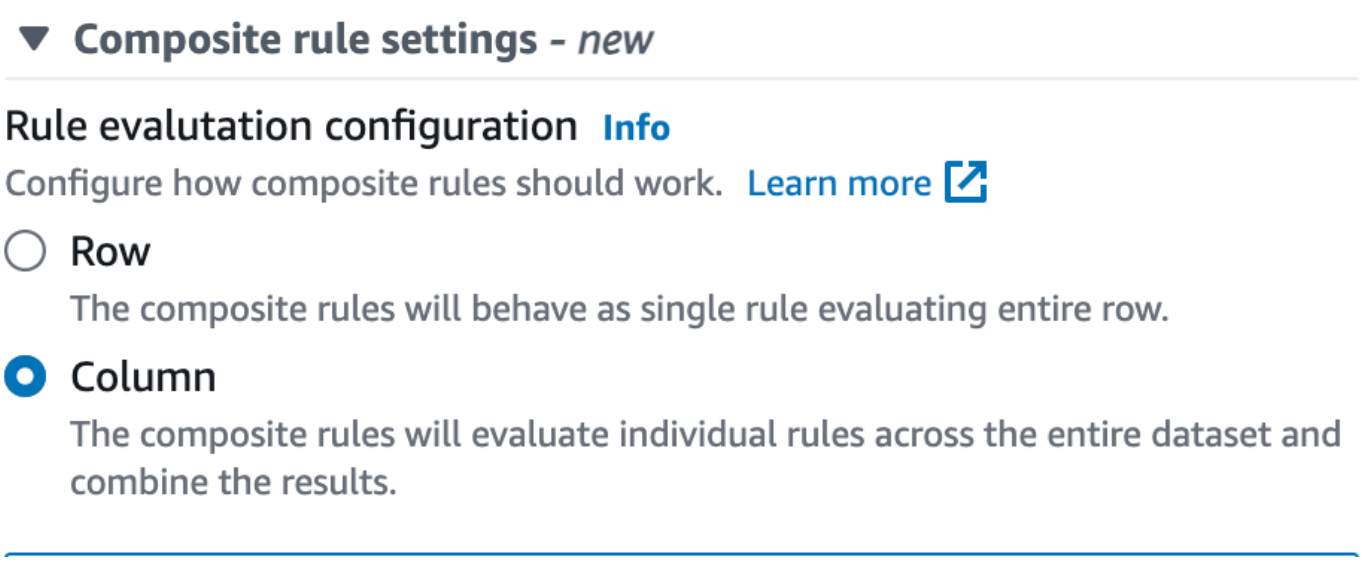
```
+-----------------------------------------------------------+-------+
|(ColumnValues "myCol1" > 1) OR (ColumnValues "myCol2" > 2)|Failed |
+-----------------------------------------------------------+-------+
```

No exemplo acima, o AWS Glue Data Quality avalia primeiro `(ColumnValues "myCol1" > 1)`, o que resultará em uma falha. Em seguida, ele avaliará `(ColumnValues "myCol2" > 2)`, que também falhará. A combinação de ambos os resultados será anotada como FALHA.

No entanto, se você preferir um comportamento semelhante ao SQL, em que a linha inteira deve ser avaliada, é necessário definir explicitamente o parâmetro `ruleEvaluation.scope` conforme mostrado em `additionalOptions` no trecho de código abaixo.

```
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"
        }
      """
    )
  )
}
```

No AWS Glue Studio e no AWS Glue Data Catalog, você pode facilmente configurar essa opção na interface do usuário, conforme mostrado abaixo.



Depois de definidas, as regras compostas se comportarão como uma única regra avaliando a linha inteira. O exemplo a seguir ilustra esse comportamento.

```
# 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
                |
+------+------+-------------------------------------------------------------
+---------------------------+
```

Algumas regras não podem ser aceitas nesse recurso porque seu resultado geral depende de limites ou proporções. Elas estão listadas abaixo.

Regras baseadas em índices:

- Completeness

- DatasetMatch
- ReferentialIntegrity
- Exclusividade

Regras dependentes de limites:

Quando as regras a seguir incluem limites, elas não são aceitas. No entanto, as regras que não envolvem `with threshold` permanecem válidas.

- ColumnDataType
- ColumnValues
- CustomSQL

Expressões

Se um tipo de regra não produzir uma resposta booliana, você deverá fornecer uma expressão como parâmetro para criar uma resposta booliana. Por exemplo, a regra a seguir verifica a média de todos os valores em uma coluna em relação a uma expressão para retornar um resultado verdadeiro ou falso.

```
Mean "colA" between 80 and 100
```

Alguns tipos de regras, como `IsUnique` e `IsComplete` já retornam uma resposta booleana.

A tabela a seguir lista as expressões que você pode usar nas regras DQDL.

Expressões DQDL compatíveis

Expressão	Descrição	Exemplo
= *x*	Será resolvido como `true` se a resposta do tipo de regra for igual a *x*.	`Completeness "colA" = "1.0", ColumnValues "colA" = "2022-06-30"`
!=*x*	x será resolvido como verdadeiro se a resposta do	`ColumnValues "colA" != "a",`

Expressão	Descrição	Exemplo
	tipo de regra for diferente de *x*.	`ColumnValues "colA" != "2022-06-30"`
> *x*	Será resolvido como `true` se a resposta do tipo de regra for maior que *x*.	`ColumnValues "colA" > 10`
< *x*	Será resolvido como `true` se a resposta do tipo de regra for menor que *x*.	`ColumnValues "colA" < 1000, ColumnValues "colA" < "2022-06-30"`
>= *x*	Será resolvido como `true` se a resposta do tipo de regra for maior que ou igual a *x*.	`ColumnValues "colA" >= 10`
<= *x*	Será resolvido como `true` se a resposta do tipo de regra for menor que ou igual a *x*.	`ColumnValues "colA" <= 1000`
between *x* and *y*	Será resolvido como `true` se a resposta do tipo de regra estiver em um intervalo especificado (exclusivo). Use esse tipo de expressão apenas para tipos numéricos e de data.	`Mean "colA" between 8 and 100, ColumnValues "colA" between "2022-05-31" and "2022-06-30"`
not between *x* and *y*	Será resolvido como verdadeiro se a resposta do tipo de regra não estiver em um intervalo especificado (inclusive). Você só deve usar esse tipo de expressão para tipos numéricos e dados.	`ColumnValues "colA" not between "2022-05-31" and "2022-06-30"`

Expressão	Descrição	Exemplo
in [*a, b, c,...*]	Será resolvido para `true` se a resposta do tipo de regra está no conjunto especific ado.	`ColumnValues "colA" in [ 1, 2, 3 ], ColumnValues "colA" in [ "a", "b", "c" ]`
not in [*a, b, c, ...*]	Será resolvido para `true` se a resposta do tipo de regra não estiver no conjunto especificado.	`ColumnValues "colA" not in [ 1, 2, 3 ], ColumnValues "colA" not in [ "a", "b", "c" ]`
matches */ab+c/i*	Será resolvido como `true` se a resposta do tipo de regra corresponder a uma expressão regular.	`ColumnValues "colA" matches "[a-zA-Z]*"`
not matches */ab+c/i*	Resolve para `true` se a resposta do tipo de regra não corresponder a uma expressão regular.	`ColumnValues "colA" not matches "[a-zA-Z]*"`
now( )	Só funciona com o tipo de regra `ColumnValues` para criar uma expressão de data.	`ColumnValues "load_date" > (now() - 3 days)`
matches/in […]/not matches/not in [...] `with threshold`	Especifica a porcentagem de valores que correspon dem às condições da regra. Funciona somente com os tipos de regras `ColumnVal ues` , `ColumnDataType` e `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`

Palavras-chave para NULL, EMPTY e WHITESPACES_ONLY

Se desejar validar se uma coluna de string tem uma string nula, vazia ou apenas com espaços em branco, você pode usar as seguintes palavras-chave:

- NULL/null: Essa palavra-chave é resolvida como verdadeira para um valor `null` em uma coluna de string.

  `ColumnValues "colA" != NULL with threshold > 0.5` retornaria verdadeiro se mais de 50% dos seus dados não tivessem valores nulos.

  `(ColumnValues "colA" = NULL) or (ColumnLength "colA" > 5)` retornaria verdadeiro para todas as linhas que têm um valor nulo ou comprimento > 5. Observe que isso exigirá o uso da opção “compositeRuleEvaluation.method” = “ROW”.
- EMPTY/empty: essa palavra-chave é resolvida como verdadeira para um valor de string vazio ("") em uma coluna de string. Alguns formatos de dados transformam nulos em uma coluna de string para strings vazias. Essa palavra-chave ajuda a filtrar strings vazias em seus dados.

  `(ColumnValues "colA" = EMPTY) or (ColumnValues "colA" in ["a", "b"])` retornaria verdadeiro se uma linha estivesse vazia, "a" ou "b". Observe que isso requer o uso da opção “compositeRuleEvaluation.method” = “ROW”.
- WHITESPACES_ONLY/whitespaces_only: essa palavra-chave é resolvida como verdadeira para uma string somente com espaços em branco (" ") em uma coluna de string.

  `ColumnValues "colA" not in ["a", "b", WHITESPACES_ONLY]` retornaria verdadeiro se uma linha não fosse "a" ou "b" nem apenas espaços em branco.

  Regras compatíveis:

  - ColumnValues

Para uma expressão numérica ou baseada em data, se você quiser validar se uma coluna tem um valor nulo, poderá usar as palavras-chave a seguir.

- NULL/null: Essa palavra-chave é resolvida como verdadeira para um valor nulo em uma coluna de string.

  `ColumnValues "colA" in [NULL, "2023-01-01"]` retornaria verdadeiro se as datas em sua coluna fossem iguais a `2023-01-01` ou nulas.

`(ColumnValues "colA" = NULL) or (ColumnValues "colA" between 1 and 9)` retornaria verdadeiro para todas as linhas que têm um valor nulo ou valores entre 1 e 9. Observe que isso exigirá o uso da opção “compositeRuleEvaluation.method” = “ROW”.

Regras compatíveis:

- ColumnValues

Filtrando com a cláusula Where

Você pode filtrar seus dados ao criar regras. Isso é útil quando você deseja aplicar regras condicionais.

```
<DQDL Rule> where "<valid SparkSQL where clause> "
```

O filtro deve ser especificado com a `where` palavra-chave, seguida por uma instrução SparkSQL válida entre aspas. `("")`

Se você quiser adicionar a cláusula where a uma regra com um limite, a cláusula where deverá ser especificada antes da condição de limite.

```
<DQDL Rule> where "valid SparkSQL statement>" with threshold <threshold condition>
```

Com essa sintaxe, você pode escrever regras como as seguintes.

```
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)"
```

Vamos validar se a instrução SparkSQL fornecida é válida. Se for inválida, a avaliação da regra falhará e lançaremos o an `IllegalArgumentException` com o seguinte formato:

```
Rule <DQDL Rule> where "<invalid SparkSQL>" has provided an invalid where clause :
<SparkSQL Error>
```

Comportamento da cláusula Where quando a identificação do registro de erro em nível de linha está ativada

Com o AWS Glue Data Quality, você pode identificar registros específicos que falharam. Ao aplicar uma cláusula where às regras que oferecem suporte a resultados em nível de linha, rotularemos as linhas que são filtradas pela cláusula where como. `Passed`

Se você preferir rotular separadamente as linhas filtradas como`SKIPPED`, defina o seguinte `additionalOptions` para a tarefa 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 exemplo, consulte a regra e o quadro de dados a seguir:

```
IsComplete att2 where "att1 = 'a'"
```

id	em 1	em 2	Resultados em nível de linha (padrão)	Resultados em nível de linha (opção ignorada)	Comentários
1	a	f	PASSARAM	PASSARAM	
2	b	d	PASSARAM	SKIPPED	A linha é filtrada, pois `att1` não é `"a"`
3	a	nulo	COM FALHA	COM FALHA	
4	a	f	PASSARAM	PASSARAM	
5	b	nulo	PASSARAM	SKIPPED	A linha é filtrada, pois `att1` não é `"a"`
6	a	f	PASSARAM	PASSARAM	

Regras dinâmicas

Agora é possível criar regras dinâmicas para comparar as métricas atuais produzidas por suas regras com seus valores históricos. Essas comparações históricas são habilitadas ao usar o operador `last()` em expressões. Por exemplo, a regra `RowCount > last()` terá êxito quando o número de linhas na execução atual for maior do que a contagem de linhas anterior mais recente para o mesmo conjunto de dados. `last()` usa um argumento opcional de número natural que descreve quantas métricas anteriores devem ser consideradas; `last(k)` em que `k >= 1` fará referência às últimas k métricas.

- Se nenhum ponto de dados estiver disponível, `last(k)` retornará o valor padrão 0,0.
- Se menos de k métricas estiverem disponíveis, `last(k)` retornará todas as métricas anteriores.

Para formar expressões válidas, use `last(k)`, em que `k > 1` requer uma função de agregação para reduzir vários resultados históricos a um único número. Por exemplo, `RowCount > avg(last(5))` verificará se a contagem de linhas do conjunto de dados atual é estritamente maior do que a média das últimas cinco contagens de linhas do mesmo conjunto de dados. `RowCount > last(5)` produzirá um erro porque a contagem de linhas do conjunto de dados atual não pode ser comparada de forma significativa a uma lista.

Funções de agregação compatíveis:

- `avg`
- `median`
- `max`
- `min`
- `sum`
- `std` (desvio padrão)
- `abs` (valor absoluto)
- `index(last(k), i)` permitirá selecionar o `i` valor mais recente entre os últimos `k`. `i` é indexado em zero, então `index(last(3), 0)` retornará o ponto de dados mais recente e `index(last(3), 3)` resultará em um erro, pois há apenas três pontos de dados e tentamos indexar o 4.º mais recente.

Exemplos de expressões

ColumnCorrelation

- `ColumnCorrelation "colA" "colB" < avg(last(10))`

DistinctValuesCount

- `DistinctValuesCount "colA" between min(last(10))-1 and max(last(10))+1`

A maioria dos tipos de regras com condições ou limites numéricos é compatível com as regras dinâmicas. Consulte a tabela fornecida, Analisadores e regras, para determinar se as regras dinâmicas são compatíveis com seu tipo de regra.

Analisadores

> **Note**
>
> Os analisadores não são compatíveis com o AWS Glue Data Catalog.

As regras de DQDL usam funções chamadas analisadores para coletar informações sobre seus dados. Essas informações são empregadas por uma expressão booleana da regra para determinar se a regra terá êxito ou não. Por exemplo, a RowCount regra `RowCount > 5` usará um analisador de contagem de linhas para descobrir o número de linhas em seu conjunto de dados e comparar essa contagem com a expressão `> 5` para verificar se existem mais de cinco linhas no conjunto de dados atual.

Às vezes, em vez de criar regras, recomendamos criar analisadores e depois fazer com que eles gerem estatísticas que possam ser usadas para detectar anomalias. Para esses casos, será necessário criar analisadores. Os analisadores diferem das regras nas seguintes maneiras:

Característica	Analisadores	Regras
Parte do conjunto de regras	Sim	Sim
Gera estatísticas	Sim	Sim
Gera observações	Sim	Sim
Pode avaliar e afirmar uma condição	Não	Sim
É possível configurar ações, como interromper os trabalhos em caso de falha ou continuar o processamento do trabalho	Não	Sim

Os analisadores podem existir de forma independente, sem a necessidade de regras, permitindo a configuração rápida e a construção progressiva de regras de qualidade de dados.

Alguns tipos de regras podem ser inseridos no bloco de `Analyzers` do seu conjunto de regras para executar as regras necessárias para os analisadores e coletar informações sem aplicar verificações

para nenhuma condição. Alguns analisadores não estão associados a regras e só podem ser inseridos no bloco de `Analyzers`. A tabela a seguir indica se cada item é compatível como uma regra ou como um analisador independente, junto com detalhes adicionais para cada tipo de regra.

Exemplo de conjunto de regras com o analisador

O seguinte conjunto de regras usa:

- uma regra dinâmica para verificar se um conjunto de dados está crescendo acima da média final nas últimas três execuções de trabalho;
- um analisador de `DistinctValuesCount` para registrar o número de valores distintos na coluna `Name` do conjunto de dados;
- um analisador de `ColumnLength` para rastrear o tamanho mínimo e máximo de `Name` ao longo do tempo.

Os resultados das métricas do analisador podem ser visualizados na guia Data Quality da execução do trabalho.

```
Rules = [
   RowCount > avg(last(3))
]
Analyzers = [
   DistinctValuesCount "Name",
   ColumnLength "Name"
]
```

Comentários

É possível usar o caractere "#" para adicionar um comentário ao seu documento DQDL. Qualquer coisa após o caractere "#" e até o final da linha é ignorada pelo DQDL.

```
Rules = [
    # More items should generally mean a higher price, so correlation should be
 positive
    ColumnCorrelation "price" "num_items" > 0
]
```

Referência de tipos de regra DQDL

Esta seção fornece uma referência para cada tipo de regra compatível com o AWS Glue Data Quality.

> **Note**
>
> - Atualmente, o DQDL não é compatível com dados de colunas aninhadas ou do tipo lista.
> - Os valores entre colchetes na tabela abaixo serão substituídos pelas informações fornecidas nos argumentos da regra.
> - As regras normalmente exigem um argumento adicional para expressão.

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatí l como regra?	Compatí l como analisad r?	Retorna resultad s em nível de linha?	Compatí l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
Aggregat Match	Verifica se há correspo dência entre dois conjunto de dados compara o as métricas resumid , como o valor total	Uma ou mais agregaçõ s	Quando os nomes da primeira e da segunda coluna de agregaçã coincider : `Column [C olumn].`	Sim	Não	Não	Não	Não	Não

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?

	das vendas. Útil para instituiç ões financeir as compara m se todos os dados são ingeridos dos sistemas de origem.		`gregate tch`
			
			Quando os nomes da primeira e da segunda coluna de agregaçã divergem
			
			`Column [C olumn1, lumn2]. gregate tch`						

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisado r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?

AllStatis tics	Analisad r indepenc nte para coletar diversas métricas para a coluna fornecida ou para todas as colunas de um conjunto de dados.	Um único nome de coluna, OU "AllColur ns”	Para as colunas de todos os tipos:
			`Dataset .RowCou`
			`Column. [C olumn]. mpleter s`
			`Column. [C olumn]. iquenes`
			Métricas adicionai s para colunas com valor de string:
				Não	Sim	Não	Não	Não	Não

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?

			`ColumnL gth metrics`
			
			Métricas adicionai s para colunas com valor numéricc
			
			`Column\ ues metrics`						

ColumnC relation	Verifica em que medida duas colunas estão correlaci onadas.	Exatame e dois nomes de coluna	`Multicc mn. [Colum n1], [Colu mn2].Cc mnCorre tion`	Sim	Sim	Não	Sim	Não	Sim

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
ColumnC nt	Verifica se alguma coluna foi descarta a.	Nenhum	`Dataset .Column unt`	Sim	Não	Não	Sim	Sim	Não
ColumnD aType	Verifica se uma coluna é compatív l com um tipo de dados.	Exatame e um nome de coluna	`Column [C olumn]. lumnDat ype.Con iance`	Sim	Não	Não	Sim, na expressã de limite em nível de linha	Não	Sim

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
ColumnE sts	Verifica se existem colunas em um conjunto de dados. Isso permite que os clientes criem plataform as de dados de autoaten imento para garantir que determin das colunas sejam disponibi lizadas.	Exatame e um nome de coluna	N/D	Sim	Não	Não	Não	Não	Não

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?

ColumnL gth	Verifica se o comprim to dos dados é consister te.	Exatame e um nome de coluna	`Column [C olumn]. ximumLe th`
			
			`Column [C olumn]. nimumLe th`
			
			Métrica adicional quando o limite de nível de linha é fornecid :
			
			`Column [C olumn]. lumnVal s.Compl nce`
				Sim	Sim	Sim, quando o limite de nível de linha é fornecid	Não	Sim. Gera apenas observaç es ao analisar o comprim to mínimo e máximo	Sim

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
ColumnN esMatch ttern	Verifica se os nomes das colunas correspo dem a padrões definidos . Útil para equipes de governar a imporem consistên cia nos nomes das colunas.	Um regex para nomes de colunas	`Dataset .Column mesPatt nMatchR io`	Sim	Não	Não	Não	Não	Não

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?

ColumnV ues	Verifica se os dados são consister tes de acordo com os valores definidos . Essa regra é compatív l com expressõ s regulares .	Exatame e um nome de coluna	`Column. [C olumn]. ximum`
			
			`Column. [C olumn]. nimum`
			
			Métrica adicional quando o limite de nível de linha é fornecid :
			
			`Column. [C olumn]. lumnVa] s.Comp] nce`
				Sim	Sim	Sim, quando o limite de nível de linha é fornecid	Não	Sim. Gera apenas observaç es ao analisar valores mínimos e máximos	Sim

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
Complet ess	Verifica se há espaços em branco ou nulos nos dados.	Exatame e um nome de coluna	`Column [C olumn]. mpleter s`	Sim	Sim	Sim	Sim	Sim	Sim

CustomS	Os clientes podem implemer ar praticam nte qualquer tipo de verificaç ão de qualidad de dados em SQL.	Uma instruçã SQL
		
		(Opciona) Um limite em nível de linha
			`Dataset .Custon L`
			
			Métrica adicional quando o limite de nível de linha é fornecid :
			
			`Dataset .Custon L.Compl nce`
				Sim	Não	Sim, quando o limite de nível de linha é fornecid	Sim	Não	Não

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
DataFres ness	Verifica se os dados estão atualizad os.	Exatame e um nome de coluna	`Column [C olumn]. taFresh ss.Comp ance`	Sim	Não	Sim	Não	Não	Sim
DatasetM tch	Compara dois conjunto de dados e identific a se eles estão sincroniz ados.	Nome de um conjunto de dados de referênci a Um mapeam o de colunas (Opciona) Colunas para verificar as correspo dências	`Dataset [ Referer Dataset ias].Da setMatc`	Sim	Não	Sim	Sim	Não	Não

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
DistinctV aluesCou t	Verifica a existênci a de valores duplicad s.	Exatame e um nome de coluna	`Column [C olumn]. stinct\ uesCour`	Sim	Sim	Sim	Sim	Sim	Sim
DetectAr malies	Verifica anomalia nas métricas relatadas de outro tipo de regra.	Um tipo de regra	Métricas relatadas pelo argumen do tipo de regra	Sim	Não	Não	Não	Não	Não
Entropia	Verifica a entropia dos dados.	Exatame e um nome de coluna	`Column [C olumn]. tropy`	Sim	Sim	Não	Sim	Não	Sim

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
IsComple e	Verifica se 100% dos dados estão completo .	Exatame e um nome de coluna	`Column [C olumn]. mpleter s`	Sim	Não	Sim	Não	Não	Sim
IsPrimary Key	Verifica se uma coluna é uma chave primária (não NULA e exclusiva).	Exatame e um nome de coluna	Para colunas únicas: `Column [C olumn]. iquenes` Para várias colunas: `Multico mn[Comm elimite olumns] niquene`	Sim	Não	Sim	Não	Não	Sim

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
IsUnique	Verifica se 100% dos dados estão exclusivo s.	Exatame e um nome de coluna	`Column [C olumn]. iquenes`	Sim	Não	Sim	Não	Não	Sim
Média	Verifica se a média atende ao limite definido.	Exatame e um nome de coluna	`Column [C olumn]. an`	Sim	Sim	Sim	Sim	Não	Sim

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisado r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
Referent alIntegri ty	Verifica se dois conjunto de dados têm integrida de referenci al.	Um ou mais nomes de coluna do conjunto de dados Um ou mais nomes de coluna do conjunto de dados de referênci a	`Column [R eferenc ataset/ as].Re ential egrity`	Sim	Não	Sim	Sim	Não	Não

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
RowCou	Verifica se a contager de registros atende a um limite.	Nenhum	`Dataset .RowCou`	Sim	Sim	Não	Sim	Sim	Sim
RowCou atch	Verifica se há correspo dência nas contager de registros entre dois conjunto de dados.	Alias do conjunto de dados de referênci a	`Dataset [ Referer Dataset ias].Ro ountMat`	Sim	Não	Não	Sim	Não	Não
Standarc eviation	Verifica se o desvio padrão atende ao limite.	Exatame e um nome de coluna	`Column. [C olumn]. andardD iation`	Sim	Sim	Sim	Sim	Não	Sim

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatív l como regra?	Compatív l como analisad r?	Retorna resultado s em nível de linha?	Compatív l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
SchemaM ch	Verifica se há correspo dência de esquema entre dois conjunto de dados.	Alias do conjunto de dados de referênci a	`Dataset [ Referer Dataset ias].Sc maMatch`	Sim	Não	Não	Sim	Não	Não
Soma	Verifica se a soma atende ao limite definido.	Exatame e um nome de coluna	`Column [C olumn]. m`	Sim	Sim	Não	Sim	Não	Sim

Ruletype	Descriçã	Argumer s	Métricas reportada s	Compatí l como regra?	Compatí l como analisad r?	Retorna resultad s em nível de linha?	Compatí l com regras dinâmica ?	Gera observaç es	Suporta a sintaxe da cláusula Where?
Exclusivi dade	Verifica se a exclusivi dade do conjunto de dados atende ao limite.	Exatame e um nome de coluna	`Column [C olumn]. iquenes`	Sim	Sim	Sim	Sim	Não	Sim
UniqueV ueRatio	Verifica se a razão de valores exclusivo s atende ao limite.	Exatame e um nome de coluna	`Column [C olumn]. iqueVa Ratio`	Sim	Sim	Sim	Sim	Não	Sim

Tópicos

- AggregateMatch
- ColumnCorrelation
- ColumnCount
- ColumnDataType

- ColumnExists
- ColumnLength
- ColumnNamesMatchPattern
- ColumnValues
- Completeness
- CustomSQL
- DataFreshness
- DatasetMatch
- DistinctValuesCount
- Entropia
- IsComplete
- IsPrimaryKey
- IsUnique
- Média
- ReferentialIntegrity
- RowCount
- RowCountMatch
- StandardDeviation
- Sum
- SchemaMatch
- Exclusividade
- UniqueValueRatio
- DetectAnomalies

AggregateMatch

Verifica a razão de agregações de duas colunas em relação a uma dada expressão. Esse tipo de regra funciona em vários conjuntos de dados. As agregações de duas colunas são avaliadas e uma razão é produzida dividindo o resultado da agregação da primeira coluna pelo resultado da agregação da segunda coluna. A razão é verificada em relação à expressão fornecida para produzir uma resposta booleana.

Sintaxe

Agregação de colunas

```
ColumnExists <AGG_OPERATION> (<OPTIONAL_REFERENCE_ALIAS>.<COL_NAME>)
```

- AGG_OPERATION: a operação a ser usada para a agregação. No momento, `sum` e `avg` são compatíveis.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- OPTIONAL_REFERENCE_ALIAS: esse parâmetro precisa ser fornecido se a coluna for de um conjunto de dados de referência e não do conjunto de dados primário. <database_name>Se você estiver usando essa regra no AWS Glue Data Catalog, seu alias de referência deverá seguir o formato ". <table_name>. <column_name>

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- COL_NAME: o nome da coluna a ser agregada.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto

Exemplo: média

```
"avg(rating)"
```

Exemplo: soma

```
"sum(amount)"
```

Exemplo: média da coluna no conjunto de dados de referência

```
"avg(reference.rating)"
```

Regra

```
AggregateMatch <AGG_EXP_1> <AGG_EXP_2> <EXPRESSION>
```

- AGG_EXP_1: a agregação da primeira coluna.

Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto

Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto

- AGG_EXP_2: a agregação da segunda coluna.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: Aggregate Match usando soma

O exemplo de regra a seguir verifica se a soma dos valores na coluna `amount` é exatamente igual à soma dos valores na coluna `total_amount`.

```
AggregateMatch "sum(amount)" "sum(total_amount)" = 1.0
```

Exemplo: Aggregate Match usando média

O exemplo de regra a seguir verifica se a média dos valores na coluna `ratings` é igual a pelo menos 90% da média dos valores na coluna `ratings` no conjunto de dados `reference`. O conjunto de dados de referência é fornecido como uma fonte de dados adicional na experiência de ETL ou do catálogo de dados.

No AWS Glue ETL, você pode usar:

```
AggregateMatch "avg(ratings)" "avg(reference.ratings)" >= 0.9
```

No AWS Glue Data Catalog, você pode usar:

```
AggregateMatch "avg(ratings)" "avg(database_name.tablename.ratings)" >= 0.9
```

Comportamento nulo

A `AggregateMatch` regra ignorará as linhas com valores NULL no cálculo dos métodos de agregação (soma/média). Por exemplo: .

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

A média da coluna `units` será (0 + 20 + 40)/3 = 20. As linhas 101 e 103 não são consideradas nesse cálculo.

ColumnCorrelation

Verifica a correlação entre duas colunas em relação a uma determinada expressão. AWS O Glue Data Quality usa o coeficiente de correlação de Pearson para medir a correlação linear entre duas colunas. O resultado é um número entre -1 e 1 que mede a força e a direção da relacão.

Sintaxe

```
ColumnCorrelation <COL_1_NAME> <COL_2_NAME> <EXPRESSION>
```

- COL_1_NAME: o nome da primeira coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- COL_2_NAME: o nome da segunda coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: correlação de colunas

O exemplo de regra a seguir verifica se o coeficiente de correlação entre as colunas `height` e `weight` tem uma correlação positiva forte (um valor de coeficiente maior que 0,8).

```
ColumnCorrelation "height" "weight" > 0.8
```

```
ColumnCorrelation "weightinkgs" "Salary" > 0.8 where "weightinkgs > 40"
```

Exemplos de regras dinâmicas

- `ColumnCorrelation "colA" "colB" between min(last(10)) and max(last(10))`
- `ColumnCorrelation "colA" "colB" < avg(last(5)) + std(last(5))`

Comportamento nulo

A `ColumnCorrelation` regra ignorará linhas com valores `NULL` no cálculo da correlação. Por exemplo: .

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

As linhas 101 e 103 serão ignoradas e `ColumnCorrelation` será 1.0.

ColumnCount

Verifica a contagem de colunas de um conjunto de dados primário em relação a uma expressão dada. Na expressão, você pode especificar o número de colunas ou um intervalo de colunas usando operadores como > e <.

Sintaxe

```
ColumnCount <EXPRESSION>
```

- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: verificação numérica da contagem de colunas

O exemplo de regra a seguir verifica se a contagem de colunas está dentro de um dado intervalo.

```
ColumnCount between 10 and 20
```

Exemplos de regras dinâmicas

- `ColumnCount >= avg(last(10))`
- `ColumnCount between min(last(10))-1 and max(last(10))+1`

ColumnDataType

Verifica o tipo de dados inerente dos valores em uma dada coluna em relação ao tipo esperado fornecido. Aceita uma expressão `with threshold` para verificar um subconjunto dos valores da coluna.

Sintaxe

```
ColumnDataType <COL_NAME> = <EXPECTED_TYPE>
   ColumnDataType <COL_NAME> = <EXPECTED_TYPE> with threshold <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: string

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- EXPECTED_TYPE: o tipo esperado dos valores da coluna.

  Valores compatíveis: booleano, data, timestamp, inteiro, duplo, flutuante, longo

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- EXPRESSION: uma expressão opcional para especificar a porcentagem de valores que devem ser do tipo esperado.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto

Exemplo: números inteiros do tipo de dados da coluna como strings

O exemplo de regra a seguir verifica se os valores da coluna da, que é do tipo string, são, na verdade, números inteiros.

```
ColumnDataType "colA" = "INTEGER"
```

Exemplo: números inteiros do tipo de dados da coluna como strings verificam um subconjunto dos valores

O exemplo de regra a seguir verifica se mais de 90% dos valores da coluna dada, que é do tipo string, são, na verdade, números inteiros.

```
ColumnDataType "colA" = "INTEGER" with threshold > 0.9
```

ColumnExists

Verifica se uma coluna existe.

Sintaxe

```
ColumnExists <COL_NAME>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: qualquer tipo de coluna

Exemplo: A coluna existe

O exemplo de regra a seguir verifica se a coluna nomeada `Middle_Name` existe.

```
ColumnExists "Middle_Name"
```

ColumnLength

Verifica se o tamanho de cada linha de uma coluna está de acordo com uma determinada expressão.

Sintaxe

```
ColumnLength <COL_NAME><EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: string
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: tamanho da linha da coluna

O exemplo de regra a seguir verifica se o valor em cada linha na coluna nomeada `Postal_Code` contém 5 caracteres.

```
ColumnLength "Postal_Code" = 5
ColumnLength "weightinkgs" = 2 where "weightinkgs > 10"
```

Comportamento nulo

A `ColumnLength` regra trata `NULL`s como strings de comprimento 0. Para uma linha `NULL`:

```
ColumnLength "Postal_Code" > 4 # this will fail
```

```
ColumnLength "Postal_Code" < 6 # this will succeed
```

O exemplo de regra composta a seguir fornece uma maneira de falhar explicitamente valores `NULL`:

```
(ColumnLength "Postal_Code" > 4) AND (ColumnValues != NULL)
```

ColumnNamesMatchPattern

Verifica se os nomes de todas as colunas no conjunto de dados primário correspondem à expressão regular dada.

Sintaxe

```
ColumnNamesMatchPattern <PATTERN>
```

- PATTERN: o padrão em relação ao qual você deseja avaliar a regra de qualidade de dados.

Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto

Exemplo: os nomes das colunas correspondem ao padrão

O exemplo de regra a seguir verifica se todas as colunas começam com o prefixo "aws_"

```
ColumnNamesMatchPattern "aws_.*"
ColumnNamesMatchPattern "aws_.*" where "weightinkgs > 10"
```

ColumnValues

Executa uma expressão em relação aos valores em uma coluna.

Sintaxe

```
ColumnValues <COL_NAME> <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: qualquer tipo de coluna
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: valores permitidos

O exemplo de regra a seguir verifica se cada valor na coluna especificada está em um conjunto de valores permitidos (inclusive nulo, vazio e strings somente com espaços em branco).

```
ColumnValues "Country" in [ "US", "CA", "UK", NULL, EMPTY, WHITESPACES_ONLY ]
ColumnValues "gender" in ["F", "M"] where "weightinkgs < 10"
```

Exemplo: expressão regular

O exemplo de regra a seguir verifica os valores em uma coluna em relação a uma expressão regular.

```
ColumnValues "First_Name" matches "[a-zA-Z]*"
```

Exemple: valores de data

O exemplo de regra a seguir verifica os valores em uma coluna de datas em relação a uma expressão regular.

```
ColumnValues "Load_Date" > (now() - 3 days)
```

Exemplo: valores numéricos

O exemplo de regra a seguir verifica se os valores da coluna correspondem a uma determinada restrição numérica.

```
ColumnValues "Customer_ID" between 1 and 2000
```

Comportamento nulo

Para todas as regras `ColumnValues` (exceto `!=` e`NOT IN`), as linhas `NULL` falharão na regra. Se a regra falhar devido a um valor nulo, o motivo da falha exibirá o seguinte:

```
Value: NULL does not meet the constraint requirement!
```

O exemplo de regra composta a seguir fornece uma maneira de permitir explicitamente valores `NULL`:

```
(ColumnValues "Age" > 21) OR (ColumnValues "Age" = NULL)
```

ColumnValues As regras negadas usando a `!= not in` sintaxe e serão passadas para `NULL` as linhas. Por exemplo: .

```
ColumnValues "Age" != 21
```

```
ColumnValues "Age" not in [21, 22, 23]
```

Os exemplos a seguir fornecem uma maneira de falhar explicitamente valores `NULL`

```
(ColumnValues "Age" != 21) AND (ColumnValues "Age" != NULL)
```

```
ColumnValues "Age" not in [21, 22, 23, NULL]
```

Completeness

Verifica a porcentagem de valores completos (não nulos) em uma coluna em relação a uma determinada expressão.

Sintaxe

```
Completeness <COL_NAME> <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: qualquer tipo de coluna
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: porcentagem de valor nulo

O exemplo de regras a seguir verifica se mais de 95 por cento dos valores em uma coluna são completos .

```
Completeness "First_Name" > 0.95
Completeness "First_Name" > 0.95 where "weightinkgs > 10"
```

Exemplos de regras dinâmicas

- `Completeness "colA" between min(last(5)) - 1 and max(last(5)) + 1`
- `Completeness "colA" <= avg(last(10))`

Comportamento nulo

Observação sobre formatos de dados CSV: linhas em branco em colunas CSV podem apresentar vários comportamentos.

- Se uma coluna for do tipo `String`, a linha em branco será reconhecida como uma string vazia e não falhará na regra `Completeness`.
- Se uma coluna for de outro tipo de dados, por exemplo, `Int`, a linha em branco será reconhecida como `NULL` e falhará na regra `Completeness`.

CustomSQL

Esse tipo de regra foi estendido para ser compatível a dois casos de uso:

- Executar uma instrução SQL personalizada em um conjunto de dados e verificar o valor de retorno em relação a uma dada expressão.
- Executar uma instrução SQL personalizada na qual você especifica um nome de coluna na instrução SELECT a qual você compara com alguma condição para obter resultados no nível de linha.

Sintaxe

```
CustomSql <SQL_STATEMENT> <EXPRESSION>
```

- SQL_STATEMENT: uma instrução SQL que retorna um único valor numérico entre aspas duplas.
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: SQL personalizado para recuperar o resultado de uma regra geral

Esse exemplo de regra usa uma instrução SQL para recuperar a contagem de registros de um conjunto de dados. A regra então verifica se a contagem de registros está entre 10 e 20.

```
CustomSql "select count(*) from primary" between 10 and 20
```

Exemplo: SQL personalizado para recuperar resultados no nível de linha

Esse exemplo de regra usa uma instrução SQL personalizada na qual você especifica um nome de coluna na instrução SELECT que você compara com alguma condição para obter resultados no nível de linha. Uma expressão de condição limite define um limite de quantos registros devem ser reprovados para que toda a regra seja reprovada. Observe que uma regra pode não conter uma condição e uma palavra-chave juntas.

```
CustomSql "select Name from primary where Age  > 18"
```

ou

```
CustomSql "select Name from primary where Age > 18" with threshold  > 3
```

> **Important**
>
> O alias `primary` representa o nome do conjunto de dados que você deseja avaliar. Quando você lida com trabalhos visuais do ETL no console, `primary` sempre representa o `DynamicFrame` que está sendo passado para a transformação `EvaluateDataQuality.apply()`. Quando você usa o AWS Glue Data Catalog para executar tarefas de qualidade de dados em uma tabela, `primary` representa a tabela.

Se você estiver no catálogo de dados do AWS Glue, também poderá usar os nomes reais das tabelas:

```
CustomSql "select count(*) from database.table" between 10 and 20
```

Você também pode unir várias tabelas para comparar diferentes elementos de dados:

```
CustomSql "select count(*) from database.table inner join database.table2 on id1 = id2"
 between 10 and 20
```

No AWS Glue ETL, o CustomSQL pode identificar registros que foram reprovados nas verificações de qualidade dos dados. Para que isso funcione, você precisará retornar registros que fazem parte da tabela primária na qual você está avaliando a qualidade dos dados. Os registros retornados como parte da consulta são considerados bem-sucedidos e os registros que não são retornados são considerados reprovados.

A regra a seguir garantirá que registros com idade < 100 sejam identificados como bem-sucedidos e que os registros acima sejam marcados como reprovados.

```
CustomSql "select id from primary where age < 100"
```

Essa regra CustomSQL será aprovada quando 50% dos registros tiverem mais de 10 anos e também identificará os registros que foram reprovados. Os registros retornados por esse CustomSQL serão considerados aprovados, enquanto os não retornados serão considerados reprovados.

```
CustomSQL "select ID, CustomerID from primary where age > 10" with threshold > 0.5
```

Observação: a regra CustomSQL falhará se você retornar registros que não estão disponíveis no conjunto de dados.

DataFreshness

Verifica a atualidade dos dados em uma coluna avaliando a diferença entre a hora atual e os valores de uma coluna de data. Você pode especificar uma expressão baseada em tempo para esse tipo de regra para garantir que os valores das colunas estejam atualizados.

Sintaxe

```
DataFreshness <COL_NAME> <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: data
- EXPRESSION: uma expressão numérica em horas ou dias. Você deve especificar a unidade de tempo na expressão.

Exemplo: atualidade dos dados

O exemplo de regras a seguir verifica a atualidade dos dados.

```
DataFreshness "Order_Date" <= 24 hours
DataFreshness "Order_Date" between 2 days and 5 days
```

Comportamento nulo

As regras `DataFreshness` falharão para linhas com valores `NULL`. Se a regra falhar devido a um valor nulo, o motivo da falha exibirá o seguinte:

```
80.00 % of rows passed the threshold
```

onde 20% das linhas que falharam incluem as linhas com `NULL`.

O exemplo de regra composta a seguir fornece uma maneira de permitir explicitamente valores `NULL`:

```
(DataFreshness "Order_Date" <= 24 hours) OR (ColumnValues "Order_Date" = NULL)
```

Atualidade dos dados para objetos do Amazon S3

Às vezes, você precisará validar a atualidade dos dados com base na hora de criação do arquivo Amazon S3. Para fazer isso, é possível usar o código a seguir para obter o carimbo de data/hora e adicioná-lo ao seu quadro de dados e, em seguida, aplicar as verificações de atualidade dos dados.

```
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

Verifica se os dados do conjunto de dados primário correspondem aos dados de um conjunto de dados de referência. Os dois conjuntos de dados são unidos usando os mapeamentos de colunas de chaves fornecidos. Mapeamentos de colunas adicionais podem ser fornecidos caso você deseje verificar a igualdade dos dados somente nessas colunas. Observe que, DataSetMatchpara funcionar, suas chaves de junção devem ser exclusivas e não devem ser NULL (devem ser uma chave primária). Se você não atender a essas condições, receberá a mensagem de erro “O mapa de teclas fornecido não é adequado para determinados quadros de dados”. Nos casos em que você não pode unir chaves exclusivas, considere usar outros tipos de regras, como AggregateMatcha correspondência em dados resumidos.

Sintaxe

```
DatasetMatch <REFERENCE_DATASET_ALIAS> <JOIN CONDITION WITH
 MAPPING> <OPTIONAL_MATCH_COLUMN_MAPPINGS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS: o alias do conjunto de dados de referência com o qual você compara os dados do conjunto de dados primário.
- KEY_COLUMN_MAPPINGS: uma lista separada por vírgulas dos nomes das colunas que formam uma chave nos conjuntos de dados. Se os nomes das colunas não forem iguais nos dois conjuntos de dados, você deverá separá-los com um `->`
- OPTIONAL_MATCH_COLUMN_MAPPINGS: você pode fornecer esse parâmetro se quiser verificar se existe correspondência de dados somente em determinadas colunas. É usada a mesma sintaxe dos mapeamentos de colunas de chaves. Se esse parâmetro não for fornecido,

compararmos os todas as colunas restantes. As colunas sem chave restantes, devem ter os mesmos nomes nos dois conjuntos de dados.

- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: comparar os conjuntos de dados usando a coluna ID

O exemplo de regra a seguir verifica se mais de 90% do conjunto de dados primário corresponde ao conjunto de dados de referência, usando a coluna "ID" para unir os dois conjuntos de dados. Ele compara todas as colunas nesse caso.

```
DatasetMatch "reference" "ID" >= 0.9
```

Exemplo: comparar os conjuntos de dados usando várias colunas de chaves

No exemplo a seguir, o conjunto de dados primário e o conjunto de dados de referência têm nomes diferentes para as colunas-chave. `ID_1` e `ID_2` juntos formam uma chave composta no conjunto de dados primário. `ID_ref1` e `ID_ref2` juntos formam uma chave composta no conjunto de dados de referência. Nesse cenário, você pode usar a sintaxe especial para fornecer os nomes das colunas.

```
DatasetMatch "reference" "ID_1->ID_ref1,ID_ref2->ID_ref2" >= 0.9
```

Exemplo: comparar conjuntos de dados usando várias colunas de chaves e verifique se a coluna específica corresponde

Este exemplo se baseia no exemplo anterior. Queremos verificar se apenas a coluna que contém os valores corresponde. Essa coluna é denominada `Amount1` no conjunto de dados primário e `Amount2` no conjunto de dados de referência. Você quer uma correspondência exata.

```
DatasetMatch "reference" "ID_1->ID_ref1,ID_ref2->ID_ref2" "Amount1->Amount2" >= 0.9
```

DistinctValuesCount

Verifica o número de valores distintos em uma coluna em relação a uma determinada expressão.

Sintaxe

```
DistinctValuesCount <COL_NAME> <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: qualquer tipo de coluna
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: contagem distinta de valores de colunas

O exemplo de regra a seguir verifica se a coluna denominada `State` contém mais de 3 valores distintos.

```
DistinctValuesCount "State" > 3
DistinctValuesCount "Customer_ID" < 6  where "Customer_ID < 10"
```

Exemplos de regras dinâmicas

- `DistinctValuesCount "colA" between avg(last(10))-1 and avg(last(10))+1`
- `DistinctValuesCount "colA" <= index(last(10),2) + std(last(5))`

Entropia

Verifica se o valor de entropia de uma coluna corresponde a uma determinada expressão. A entropia mede o nível de informação que está contida em uma mensagem. Dada a distribuição de probabilidade pelos valores em uma coluna, a entropia descreve quantos bits são necessários para identificar um valor.

Sintaxe

```
Entropy <COL_NAME> <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: qualquer tipo de coluna
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: entropia de coluna

O exemplo de regra a seguir verifica se a coluna determinada `Feedback` tem um valor de entropia maior que um.

```
Entropy "Star_Rating" > 1
Entropy "First_Name" > 1  where "Customer_ID < 10"
```

Exemplos de regras dinâmicas

- `Entropy "colA" < max(last(10))`
- `Entropy "colA" between min(last(10)) and max(last(10))`

IsComplete

Verifica se todos os valores em uma coluna estão completos (não nulos).

Sintaxe

```
IsComplete <COL_NAME>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: qualquer tipo de coluna

Exemplo: valores nulos

O exemplo a seguir verifica se todos os valores em uma coluna denominada `email` não são 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"
```

Comportamento nulo

Observação sobre formatos de dados CSV: linhas em branco em colunas CSV podem apresentar vários comportamentos.

- Se uma coluna for do tipo `String`, a linha em branco será reconhecida como uma string vazia e não falhará na regra `Completeness`.
- Se uma coluna for de outro tipo de dados, por exemplo, `Int`, a linha em branco será reconhecida como `NULL` e falhará na regra `Completeness`.

IsPrimaryKey

Verifica se uma coluna contém uma chave primária. Uma coluna contém uma chave primária se todos os valores na coluna forem exclusivos e completos (não nulos).

Sintaxe

```
IsPrimaryKey <COL_NAME>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: qualquer tipo de coluna

Exemplo: chave primária

O exemplo de regra a seguir verifica se a coluna denominada `Customer_ID` contém uma chave primária.

```
IsPrimaryKey "Customer_ID"
IsPrimaryKey "Customer_ID" where "Customer_ID < 10"
```

Exemplo: chave primária com várias colunas. Qualquer um dos exemplos a seguir é válido.

```
IsPrimaryKey "colA" "colB"
IsPrimaryKey "colA" "colB" "colC"
IsPrimaryKey colA "colB" "colC"
```

IsUnique

Verifica se todos os valores em uma coluna são exclusivos e retorna um valor booliano.

Sintaxe

```
IsUnique <COL_NAME>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: qualquer tipo de coluna

Exemplo: valores de coluna exclusivos

O exemplo de regra a seguir verifica se todos os valores em uma coluna denominada `email` são exclusivos.

```
IsUnique "email"
IsUnique "Customer_ID" where "Customer_ID < 10"]
```

Média

Verifica se o valor mediano (média) de uma coluna corresponde a uma determinada expressão.

Sintaxe

```
Mean <COL_NAME> <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: valor médio

O exemplo de regra a seguir verifica se a média de todos os valores em uma coluna excede um limite.

```
Mean "Star_Rating" > 3
Mean "Salary" < 6200 where "Customer_ID < 10"
```

Exemplos de regras dinâmicas

- `Mean "colA" > avg(last(10)) + std(last(2))`
- `Mean "colA" between min(last(5)) - 1 and max(last(5)) + 1`

Comportamento nulo

A regra `Mean` ignorará linhas com valores `NULL` no cálculo da média. Por exemplo: .

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

A média da coluna `units` será (0 + 20 + 40)/3 = 20. As linhas 101 e 103 não são consideradas nesse cálculo.

ReferentialIntegrity

Verifica até que ponto os valores de um conjunto de colunas no conjunto de dados primário são um subconjunto dos valores de um conjunto de colunas em um conjunto de dados de referência.

Sintaxe

```
ReferentialIntegrity <PRIMARY_COLS> <REFERENCE_DATASET_COLS> <EXPRESSION>
```

- PRIMARY_COLS: uma lista de nomes de coluna separados por vírgulas no conjunto de dados primário.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- REFERENCE_DATASET_COLS: esse parâmetro contém duas partes separadas por um ponto. A primeira parte é o alias do conjunto de dados de referência. A segunda parte é a lista separada por vírgulas dos nomes das colunas do conjunto de dados de referência entre colchetes.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto

- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: verificar a integridade referencial de uma coluna de códigos postais

O exemplo de regra a seguir verifica se mais de 90% dos valores da coluna `zipcode` do conjunto de dados primário estão presentes na coluna `zipcode` do conjunto de dados `reference`.

```
ReferentialIntegrity "zipcode" "reference.zipcode" >= 0.9
```

Exemplo: verificar a integridade referencial das colunas de cidades e estados

No exemplo a seguir, existem colunas contendo informações sobre cidades e estados no conjunto de dados primário e no conjunto de dados de referência. Os nomes das colunas são diferentes nos dois conjuntos de dados. A regra verifica se o conjunto de valores das colunas do conjunto de dados primário é exatamente igual ao conjunto de valores das colunas do conjunto de dados de referência.

```
ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" = 1.0
```

Exemplos de regras dinâmicas

- `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

Verifica a contagem de linhas de um conjunto de dados em relação a uma determinada expressão. Na expressão, você pode especificar o número de linhas ou um intervalo de linhas usando operadores como > e <.

Sintaxe

```
RowCount <EXPRESSION>
```

- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: verificação numérica da contagem de linhas

O exemplo de regra a seguir verifica se a contagem de linhas está dentro de um determinado intervalo.

```
RowCount between 10 and 100
RowCount between 1 and 50 where "Customer_ID < 10"
```

Exemplos de regras dinâmicas

```
RowCount > avg(lats(10)) *0.8
```

RowCountMatch

Verifica a razão entre a contagem de linhas do conjunto de dados primário e a contagem de linhas de um conjunto de dados de referência em relação à expressão dada.

Sintaxe

```
RowCountMatch <REFERENCE_DATASET_ALIAS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS O alias do conjunto de dados de referência com o qual comparar as contagens de linhas.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: verificação da contagem de linhas em relação a um conjunto de dados de referência

O exemplo de regra a seguir verifica se a contagem de linhas do conjunto de dados primário é pelo menos 90% da contagem de linhas do conjunto de dados de referência.

```
RowCountMatch "reference" >= 0.9
```

StandardDeviation

Verifica o desvio padrão de todos os valores em uma coluna em relação a uma determinada expressão.

Sintaxe

```
StandardDeviation <COL_NAME> <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: desvio padrão

O exemplo de regra a seguir verifica se o desvio padrão dos valores em uma coluna denominada `colA` é menor que um valor especificado.

```
StandardDeviation "Star_Rating" < 1.5
StandardDeviation "Salary" < 3500 where "Customer_ID < 10"
```

Exemplos de regras dinâmicas

- `StandardDeviation "colA" > avg(last(10) + 0.1`
- `StandardDeviation "colA" between min(last(10)) - 1 and max(last(10)) + 1`

Comportamento nulo

A regra `StandardDeviation` ignorará linhas com valores `NULL` no cálculo do desvio padrão. Por exemplo: .

```
+---+-----------+-----------+
|id |units1     |units2     |
+---+-----------+-----------+
|100|0          |0          |
|101|null       |0          |
|102|20         |20         |
|103|null       |0          |
|104|40         |40         |
+---+-----------+-----------+
```

O desvio padrão da coluna `units1` não considerará as linhas 101 e 103 e resultará em 16,33. O desvio padrão da coluna `units2` resultará em 16.

Sum

Verifica a soma de todos os valores em uma coluna em relação a uma determinada expressão.

Sintaxe

```
Sum <COL_NAME> <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: soma

O exemplo de regra a seguir verifica se a soma de todos os valores em uma coluna excede um limite.

```
Sum "transaction_total" > 500000
Sum "Salary" < 55600 where "Customer_ID < 10"
```

Exemplos de regras dinâmicas

- `Sum "ColA" > avg(last(10))`
- `Sum "colA" between min(last(10)) - 1 and max(last(10)) + 1`

Comportamento nulo

A regra `Sum` ignorará linhas com valores `NULL` no cálculo da soma. Por exemplo: .

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
```

```
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

A soma da coluna `units` não considerará as linhas 101 e 103 e resultará em (0 + 20 + 40) = 60.

SchemaMatch

Verifica se o esquema do conjunto de dados primário corresponde ao esquema de um conjunto de dados de referência. A verificação de esquema é feita coluna por coluna. O esquema de duas colunas corresponde se os nomes forem idênticos e os tipos forem idênticos. A ordem das colunas não importa.

Sintaxe

```
SchemaMatch <REFERENCE_DATASET_ALIAS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS: o alias do conjunto de dados de referência com o qual comparar os esquemas.

  Tipos de coluna compatíveis: byte, decimal, duplo, flutuante, inteiro, longo, curto
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: SchemaMatch

A regra de exemplo a seguir verifica se o esquema do conjunto de dados primário corresponde exatamente ao esquema de um conjunto de dados de referência.

```
SchemaMatch "reference" = 1.0
```

Exclusividade

Verifica a porcentagem de valores exclusivos em uma coluna em relação a uma determinada expressão. Valores exclusivos ocorrem exatamente uma vez.

Sintaxe

```
Uniqueness <COL_NAME> <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: qualquer tipo de coluna
- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: porcentagem de exclusividade

O exemplo de regra a seguir verifica se a porcentagem de valores exclusivos em uma coluna corresponde a determinados critérios numéricos.

```
Uniqueness "email" = 1.0
Uniqueness "Customer_ID" != 1.0 where "Customer_ID < 10"
```

Exemplos de regras dinâmicas

- `Uniqueness "colA" between min(last(10)) and max(last(10))`
- `Uniqueness "colA" >= avg(last(10))`

UniqueValueRatio

Verifica a razão de valores exclusivos em uma coluna em relação a uma determinada expressão. Uma razão de valor exclusivo é a fração de valores exclusivos dividida pelo número de todos os valores distintos em uma coluna. Valores exclusivos ocorrem exatamente uma vez, enquanto valores distintos ocorrem pelo menos uma vez.

Por exemplo, o conjunto `[a, a, b]` contém um valor exclusivo (b) e dois valores distintos (a e b). Portanto, a proporção de valores exclusivos do conjunto é ½ = 0,5.

Sintaxe

```
UniqueValueRatio <COL_NAME> <EXPRESSION>
```

- COL_NAME: o nome da coluna em relação à qual você deseja avaliar a regra de qualidade de dados.

  Tipos de coluna compatíveis: qualquer tipo de coluna

- EXPRESSION: uma expressão a ser executada na resposta do tipo de regra para produzir um valor booliano. Para ter mais informações, consulte Expressões.

Exemplo: razão de valores exclusivos

Esse exemplo verifica a proporção de valores exclusivos de uma coluna em relação a um intervalo de valores.

```
UniqueValueRatio "test_score" between 0 and 0.5
UniqueValueRatio "Customer_ID" between 0 and 0.9 where "Customer_ID < 10"
```

Exemplos de regras dinâmicas

- `UniqueValueRatio "colA" > avg(last(10))`
- `UniqueValueRatio "colA" <= index(last(10),2) + std(last(5))`

DetectAnomalies

Detecta anomalias em uma determinada regra de qualidade de dados. Cada execução de uma DetectAnomalies regra resulta em salvar o valor avaliado para a regra dada. Quando há dados suficientes coletados, o algoritmo de detecção de anomalias pega todos os dados históricos dessa regra e executa a detecção de anomalias. DetectAnomalies a regra falha quando a anomalia é detectada. Para obter mais informações sobre qual anomalia foi detectada, consulte Observações.

Sintaxe

```
DetectAnomalies <RULE_NAME> <RULE_PARAMETERS>
```

`RULE_NAME`: o nome da regra para a qual você quer avaliar e detectar anomalias. Regras compatíveis:

- "RowCount"
- "Completeness"
- "Uniqueness"
- "Mean"
- "Sum"

- "StandardDeviation"
- "Entropy"
- "DistinctValuesCount"
- "UniqueValueRatio"
- "ColumnLength"
- "ColumnValues"
- "ColumnCorrelation"

`RULE_PARAMETERS`: algumas regras exigem parâmetros adicionais para serem executadas. Consulte a documentação da regra fornecida para ver os parâmetros necessários.

Exemplo: Anomalias para RowCount

Por exemplo, se quisermos detectar RowCount anomalias, fornecemos RowCount como regra o nome.

```
DetectAnomalies "RowCount"
```

Exemplo: Anomalias para ColumnLength

Por exemplo, se quisermos detectar ColumnLength anomalias, fornecemos ColumnLength como regra o nome e o nome da coluna.

```
DetectAnomalies "ColumnLength" "id"
```

Usar APIs para medir e gerenciar a qualidade dos dados

Este tópico descreve como usar as APIs para medir e gerenciar a qualidade dos dados.

Sumário

Pré-requisitos

- Certifique-se de que sua versão do boto3 esteja atualizada para incluir a API do AWS Glue Data Quality mais recente.
- Certifique-se de que sua versão da AWS CLI esteja atualizada, para incluir a CLI mais recente.

Se você estiver usando um trabalho do AWS Glue para executar essas APIs, poderá usar a opção a seguir para atualizar a biblioteca do boto3 para a versão mais recente:

```
—additional-python-modules boto3==<version>
```

Trabalhar com as recomendações do AWS Glue Data Quality

Para iniciar uma recomendação do AWS Glue Data Quality, execute:

```
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 uma execução de recomendação, você pode usar `pushDownPredicates` ou `catalogPartitionPredicates` para melhorar a performance e executar recomendações somente em partições específicas das fontes do 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 obter os resultados de uma recomendação do AWS Glue Data Quality, execute:

```
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
```

Do objeto de resposta acima, você pode extrair o conjunto de regras recomendado pela execução, para usar em outras etapas:

```
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 obter uma lista de todas as suas execuções de recomendação que podem ser filtradas e listadas:

```
response = client.list_data_quality_rule_recommendation_runs(
```

```
    Filter={
        'DataSource': {
            'GlueTable': {
                'DatabaseName': '<database_name>',
                'TableName': '<table_name>'
            }
        }
)
```

Para cancelar as tarefas existentes de recomendação do AWS Glue Data Quality:

```
response = client.cancel_data_quality_rule_recommendation_run(
    RunId='dqrun-d4b6b01957fdd79e59866365bf9cb0e40fxxxxxxx'
)
```

Trabalhar com conjuntos de regras do AWS Glue Data Quality

Para criar um conjunto de regras do AWS Glue Data Quality:

```
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 criar um conjunto de regras de qualidade de dados:

```
response = client.get_data_quality_ruleset(
    Name='<ruleset_name>'
)
print(response)
```

Você pode usar essa API para extrair o conjunto de regras:

```
print(response['Ruleset'])
```

Para listar todos os conjuntos de regras de qualidade de dados para uma tabela:

```
response = client.list_data_quality_rulesets()
```

Você pode usar a condição de filtro na API para filtrar todos os conjuntos de regras anexados a um banco de dados ou a uma tabela específica:

```
response = client.list_data_quality_rulesets(
    Filter={
        'TargetTable': {
            'TableName': '<table_name>',
            'DatabaseName': '<database_name>'
        }
    },
)
```

Para atualizar um conjunto de regras de qualidade de dados:

```
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 excluir um conjunto de regras de qualidade de dados:

```
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
```

Trabalhar com execuções do AWS Glue Data Quality

Para iniciar uma execução do AWS Glue Data Quality:

```
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']
```

Lembre-se de que você pode passar um parâmetro `pushDownPredicate` ou `catalogPartitionPredicate` para garantir que sua execução de qualidade de dados só tenha como objetivo um conjunto específico de partições na tabela do catálogo. Por exemplo:

```
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 obter informações sobre uma execução do AWS Glue Data Quality:

```
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 obter os resultados de uma execução do AWS Glue Data Quality:

Para uma determinada execução do AWS Glue Data Quality, você pode extrair os resultados da avaliação da execução usando o seguinte 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 listar todas as suas execuções do AWS Glue Data Quality:

```
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
```

Você pode modificar a cláusula de filtro para só mostrar os resultados entre determinadas horas ou os resultados de execuções em tabelas específicas.

Para interromper uma execução contínua do AWS Glue Data Quality:

```
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
```

Trabalhar com resultados do AWS Glue Data Quality

Para obter os resultados de uma execução do AWS Glue Data Quality:

```
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 as tarefas existentes de recomendação do AWS Glue Data Quality:

Fornecido um ID de execução do AWS Glue Data Quality, você pode extrair o ID do resultado para então obter os resultados reais, conforme mostrado abaixo:

```
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'])
```

Configurar alertas, implantações e agendamentos

Este tópico descreve como configurar alertas, implantações e agendamento para o AWS Glue Data Quality.

Sumário

Configurando alertas e notificações na EventBridge integração com a Amazon

AWS O Glue Data Quality suporta a publicação de EventBridge eventos, que são emitidos após a conclusão de uma execução de avaliação do conjunto de regras de qualidade de dados. Com isso, você pode facilmente configurar alertas quando as regras de qualidade de dados não forem atendidas.

Aqui está um exemplo de evento quando você avalia os conjuntos de regras de qualidade de dados no catálogo de dados. Com essas informações, você pode revisar os dados disponibilizados na

Amazon EventBridge. Você pode emitir chamadas de API adicionais para obter mais detalhes. Por exemplo, chame a API `get_data_quality_result` com o ID do resultado para obter os detalhes de uma execução específica.

```
{
    "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
    }
}
```

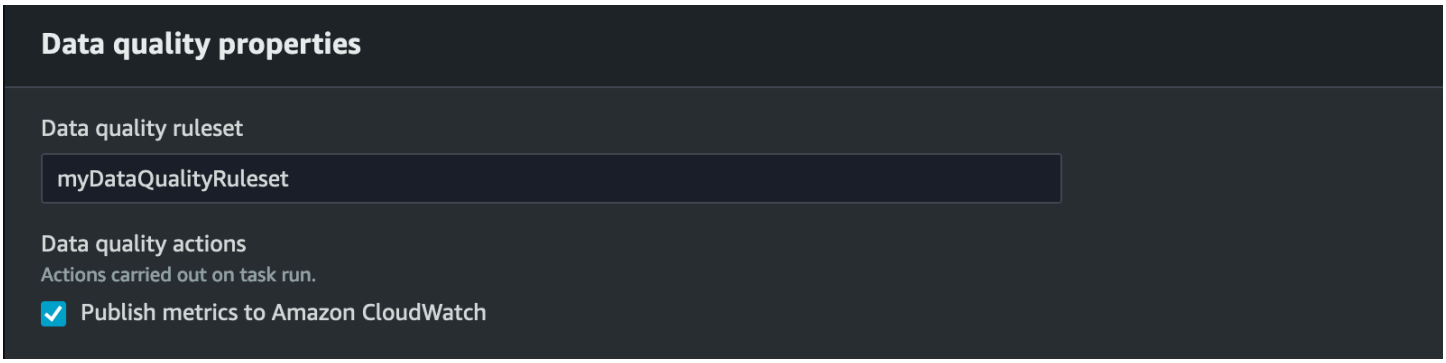
Aqui está um exemplo de evento que é publicado quando você avalia conjuntos de regras de qualidade de dados nos notebooks AWS Glue ETL ou AWS Glue Studio.

```
{
    "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 a avaliação da qualidade de dados seja executada no catálogo de dados e nas tarefas de ETL, a Amazon CloudWatch opção Publicar métricas para, selecionada por padrão, deve permanecer selecionada para que a EventBridge publicação funcione.

Configurando EventBridge notificações



Para receber os eventos emitidos e definir metas, você deve configurar EventBridge as regras da Amazon. Para criar regras:

1. Abra o EventBridge console da Amazon.
2. Escolha Regras na seção Barramentos da barra de navegação.
3. Selecione Create Rule.
4. Em Definir detalhes da regra:
   a. Em Nome, insira `myDQRule`.

b. Insira uma descrição (opcional).

c. Para barramento de eventos, selecione seu barramento de eventos. Se você não tiver um, deixe o padrão.

d. Em Tipo de regra, escolha Regra com um padrão de eventos.

5. Em Criar padrão de eventos:

a. Para a fonte do evento, selecione AWS eventos ou eventos de EventBridge parceiros.

b. Pule a seção de exemplo de evento.

c. Como método de criação, selecione Usar formulário de padrão.

d. Para padrão de eventos:

i. Selecione Serviços da AWS para Fonte do evento.

ii. Selecione Glue Data Quality para AWS atendimento.

iii. Selecione Resultados da avaliação de qualidade de dados disponíveis para Tipo de evento.

iv. Selecione FALHA para Estados específicos. Em seguida, você verá um padrão de evento como este:

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "state": ["FAILED"]
  }
}
```

v. Para obter mais opções de conexão, consulte Opções de configuração adicionais para o padrão de eventos.

6. Em Selecionar alvos:

a. Em Tipos de destino, selecione Serviço da AWS .

b. Use o menu suspenso Selecionar um destino para escolher o AWS serviço desejado ao qual se conectar (SNS, Lambda, SQS etc.) e escolha Avançar.

7. Em Configurar tag(s), clique em Adicionar novas tags para adicionar tags opcionais e escolha Avançar.

8. Você verá uma página de resumo de todas as seleções. Escolha Criar regra na parte inferior.

Opções de configuração adicionais para o padrão de eventos

Além de filtrar o evento baseado em sucesso ou fracasso, talvez você queira filtrar ainda mais os eventos com base em outros parâmetros.

Para fazer isso, vá para a seção Padrão de eventos e selecione Editar padrão para especificar parâmetros adicionais. Observe que os campos no padrão de eventos diferenciam maiúsculas de minúsculas. Estes são exemplos de configuração do padrão de eventos.

Para capturar eventos de uma tabela específica avaliando conjuntos de regras específicos, use esse tipo de padrão:

```
{
  "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 trabalhos específicos na experiência de ETL, use esse tipo de padrão:

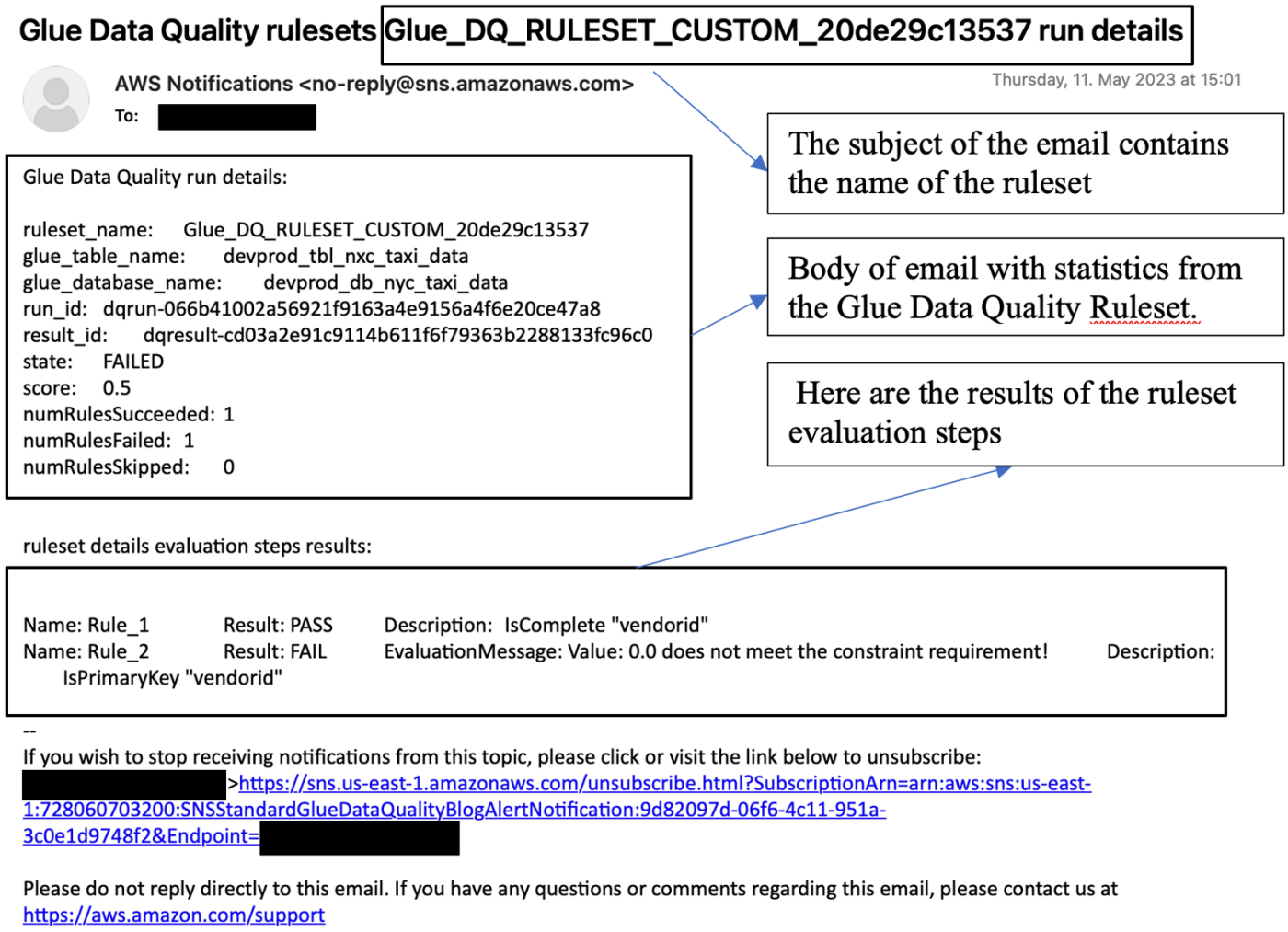
```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "context": {
      "contextType": ["GLUE_JOB"],
      "jobName": ["dq_evaluation_job1", "dq_evaluation_job2"]
    },
    "state": ["FAILED"]
  }
}
```

Para capturar eventos com uma pontuação abaixo de um limite específico (por exemplo, 70%):

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "score": [{
      "numeric": ["<=", 0.7]
    }]
  }
}
```

Formatar notificações como e-mails

Às vezes, você precisa enviar uma notificação por e-mail bem formatada para as equipes da sua empresa. Você pode usar a Amazon EventBridge e a AWS Lambda para conseguir isso.



O exemplo de código a seguir pode ser usado para formatar as notificações de qualidade de dados para gerar e-mails.

```
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 e notificações na CloudWatch integração

Nossa abordagem recomendada é configurar alertas de qualidade de dados usando a Amazon EventBridge, porque a Amazon EventBridge exige uma configuração única para alertar os clientes. No entanto, alguns clientes preferem a Amazon CloudWatch devido à familiaridade. Para esses clientes, oferecemos integração com a Amazon CloudWatch.

Cada avaliação do AWS Glue Data Quality emite um par de métricas denominadas `glue.data.quality.rules.passed` (indicando o número de regras aprovadas) e `glue.data.quality.rules.failed` (indicando o número de regras que falharam) por execução de qualidade de dados. Você pode usar essa métrica emitida para criar alarmes para avisar os usuários se uma determinada execução de qualidade de dados ficar abaixo de um limite. Para começar a configurar um alarme que enviaria um e-mail por meio de uma notificação do Amazon SNS, siga as etapas abaixo:

Para começar a configurar um alarme que enviaria um e-mail por meio de uma notificação do Amazon SNS, siga as etapas abaixo:

1. Abra o CloudWatch console da Amazon.

2. Escolha Todas as métricas em Métricas. Você verá um namespace adicional em Namespaces personalizados intitulado Glue Data Quality.

> **Note**
>
> Ao iniciar uma execução do AWS Glue Data Quality, certifique-se de que a CloudWatch caixa de seleção Publicar métricas na Amazon esteja ativada. Caso contrário, as métricas dessa execução específica não serão publicadas na Amazon CloudWatch.

   No namespace `Glue Data Quality`, você pode ver as métricas que estão sendo emitidas por tabela, por conjunto de regras. Para a finalidade deste tópico, usaremos a regra `glue.data.quality.rules.failed` e o alarme se esse valor ultrapassar 1 (indicando que, se observarmos um número de avaliações de regras com falha maior do que 1, queremos ser notificados).
3. Para criar o alarme, escolha Todos os alarmes em Alarmes.
4. Selecione Criar alarme.
5. Escolha Selecionar métrica.
6. Selecione a métrica `glue.data.quality.rules.failed` correspondente à tabela que você criou e depois escolha Selecionar métrica.
7. Na guia Especificar métricas e condições, na seção Métricas:
   a. Em Estatística, selecione Soma.
   b. Em Período, escolha 1 minuto.
8. Na seção Condições:
   a. Em Tipo de limite, escolha Estático.
   b. para Sempre que glue.data.quality.rules.failed for..., selecione Maior/Igual.
   c. Por do que... , insira 1 como o valor limite.

   Essas seleções implicam que, se a métrica `glue.data.quality.rules.failed` emitir um valor maior ou igual a 1, acionaremos um alarme. Porém, se não houver dados, nós a trataremos como aceitável.
9. Escolha Próximo.
10. Em Configurar ações:
    a. Em Gatilho de estado do alarme, escolha Em alarme.

b. Em Enviar uma notificação para o seguinte tópico do SNS, escolha Criar um novo tópico para enviar uma notificação por um novo tópico do SNS.

c. Em Endpoints de e-mail que receberão a notificação, insira o endereço de e-mail. Depois clique em Criar tópico.

d. Escolha Avançar.

11. Em Nome secreto, insira `myFirstDQAlarm` e escolha Avançar.

12. Você verá uma página de resumo de todas as seleções. Escolha Criar alarme na parte inferior.

Agora você pode ver o alarme sendo criado no painel de CloudWatch alarmes da Amazon.

Consultar resultados de qualidade de dados para criar painéis

Talvez você queira criar um painel para exibir seus resultados de qualidade de dados. Há duas maneiras de fazer isso:

Configure a Amazon EventBridge com o seguinte código para gravar os dados no 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')
    }
```

Depois de gravar no Amazon S3, você pode usar os rastreadores AWS Glue para se registrar no Athena e consultar as tabelas.

Configure um local no Amazon S3 durante uma avaliação de qualidade de dados:

Ao executar tarefas de qualidade de dados no AWS Glue Data Catalog ou no AWS Glue ETL, você pode fornecer um local do Amazon S3 para gravar os resultados de qualidade de dados no Amazon S3. Você pode usar a sintaxe abaixo para criar uma tabela referenciando o destino para o qual ler os resultados de qualidade dos dados.

Observe que você deve executar as consultas `CREATE EXTERNAL TABLE` e `MSCK REPAIR TABLE` separadamente.

```
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>;
```

Depois de criar a tabela acima, você poderá executar consultas analíticas usando o Amazon Athena.

Implantando regras de qualidade de dados usando AWS CloudFormation

Você pode usar AWS CloudFormation para criar regras de qualidade de dados. Para obter mais informações, consulte AWS CloudFormation AWS Glue.

Agendar regras de qualidade de dados

Você pode agendar as regras de qualidade de dados usando os seguintes métodos:

- Agende regras de qualidade de dados do Catálogo de Dados: nenhum usuário de código pode usar essa opção para agendar facilmente suas verificações de qualidade de dados. AWS A Glue Data Quality criará o cronograma na Amazon EventBridge. Para agendar regras de qualidade de dados:
  - Navegue até o conjunto de regras e clique em Executar.
  - Na Frequência de execução, selecione a agenda desejada e forneça um Nome de tarefa. Esse nome de tarefa é o nome da sua agenda em EventBridge.
- Use Amazon EventBridge e AWS Step Functions para orquestrar avaliações e recomendações para regras de qualidade de dados.

Solucionando erros de qualidade de dados do AWS Glue

Se você encontrar erros no AWS Glue Data Quality, use as soluções a seguir para ajudá-lo a encontrar a origem dos problemas e corrigi-los.

Sumário

- Erro: tabelas com caracteres especiais
- Erro: erro de estouro com um conjunto de regras grande
- Erro: o status geral da regras é com falha
- AnalysisException: Não é possível verificar a existência do banco de dados padrão
- Mensagem de erro: o mapa de chaves fornecido não é adequado para os quadros de dados fornecidos
- Exceção na classe de usuário: java.lang. RuntimeException : falha na busca de dados. Verifique os logins CloudWatch para obter mais detalhes
- ERRO DE INICIALIZAÇÃO: erro ao baixar o S3 para o bucket
- InvalidInputException (status: 400): DataQuality as regras não podem ser analisadas
- Erro: o Eventbridge não está acionando trabalhos do Glue DQ com base na agenda que eu configurei
- Erros de CustomSQL
- Regras dinâmicas
- Exceção na classe de usuário: org.apache.spark.sql. AnalysisException: org.apache.hadoop.hive.ql.metadata. HiveException
- UNCLASSIFIED_ERROR; IllegalArgumentException: Erro de análise: nenhuma regra ou analisador fornecido., nenhuma alternativa viável na entrada

Erro: falta do módulo AWS Glue Data Quality

Mensagem de erro: nenhum módulo denominado 'awsgluedq'.

Resolução: Esse erro ocorre quando você executa o AWS Glue Data Quality em uma versão não suportada. AWS O Glue Data Quality é suportado somente na versão 3.0 e posterior do Glue.

Erro: permissões insuficientes AWS do Lake Formation

Mensagem de erro: Exceção na classe de usuário:`com.amazonaws.services.glue.model.AccessDeniedException`: Permissões insuficientes do Lake Formation em impact_sdg_involvement (Serviço:; Código de status: 400; Código de erro:AWS Glue; ID da solicitação: AccessDeniedException 465ae693-b7ba-4df0-a4e4-6b17xxxxxxxx; Proxy: null).

Resolução: Você deve fornecer permissões suficientes no AWS Lake Formation.

Erro: os conjuntos de regras não têm nomes exclusivos

Mensagem de erro: Exceção na classe de usuário:... services.glue.model. AlreadyExistsException: já existe outro conjunto de regras com o mesmo nome.

Resolução: os conjuntos de regras precisam ser exclusivos.

Erro: tabelas com caracteres especiais

Mensagem de erro: Exceção na classe de usuário: org.apache.spark.sql. AnalysisException: não é possível resolver “C” dadas as colunas 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]; linha 1 pos 44;.

Resolução: Há uma limitação atual de que o AWS Glue Data Quality não pode ser executado em tabelas com caracteres especiais, como “.”.

Erro: erro de estouro com um conjunto de regras grande

Mensagem de erro: Exceção na classe de usuário: java.lang. StackOverflowError.

Resolução: se você tiver um conjunto de regras grande com mais de 2 mil regras, poderá encontrar esse problema. Divida as regras em vários conjuntos de regras.

Erro: o status geral da regras é com falha

Condição de erro: meu conjunto de regras teve sucesso, mas o status geral de regras é com falha.

Resolução: Esse erro provavelmente ocorreu porque você escolheu a opção de publicar métricas na Amazon CloudWatch durante a publicação. Se seu conjunto de dados estiver em uma VPC, sua VPC pode não permitir que o AWS Glue publique métricas na Amazon. CloudWatch Nesse caso, você precisa configurar um endpoint para que sua VPC acesse a Amazon. CloudWatch

AnalysisException: Não é possível verificar a existência do banco de dados padrão

Condição de erro AnalysisException: Não é possível verificar a existência do banco de dados padrão: com.amazonaws.services.glue.model. AccessDeniedException: Permissões insuficientes do Lake Formation por padrão (Serviço:AWS Glue; Código de status: 400; Código de erro:; ID da solicitação: XXXXXXXX-XXXX-XXXX-XXXXXX-XXXXXXXXXXXX AccessDeniedException; Proxy: null)

Resolução: na integração do catálogo do trabalho do AWS Glue, o AWS Glue sempre tenta verificar se o banco de dados padrão existe ou não está usando a `GetDatabase API` do AWS Glue. Quando a permissão do `DESCRIBE` Lake Formation não é concedida, ou a permissão `GetDatabase IAM` é concedida e depois o trabalho falha ao verificar a existência do banco de dados padrão.

Para resolver:

1. Adicione a permissão `DESCRIBE` no Lake Formation para o banco de dados padrão.
2. Configure o perfil do IAM anexado ao trabalho do AWS Glue como criador de banco de dados no Lake Formation. Isso criará automaticamente um banco de dados padrão e concederá ao perfil as permissões necessárias do Lake Formation.
3. Desabilite a opção `--enable-data-catalog`. (É mostrado como Usar o Data Catalog como metastore do Hive no AWS Glue Studio).

   Se você não precisar da integração do Data Catalog com o Spark SQL no trabalho, poderá desabilitá-la.

Mensagem de erro: o mapa de chaves fornecido não é adequado para os quadros de dados fornecidos

Condição de erro: o mapa de chaves fornecido não é adequado para os quadros de dados fornecidos.

Resolução: Você está usando o DataSetMatchtipo de regra e as teclas de junção têm duplicatas. Suas chaves de união devem ser exclusivas e não devem ser NULL. Nos casos em que você não pode ter chaves de junção exclusivas, considere usar outros tipos de regras, como AggregateMatcha correspondência em dados resumidos.

Exceção na classe de usuário: java.lang. RuntimeException : falha na busca de dados. Verifique os logins CloudWatch para obter mais detalhes

Condição de erro: exceção na classe de usuário: java.lang. RuntimeException : falha na busca de dados. Verifique os logins CloudWatch para obter mais detalhes.

Resolução: Isso acontece quando você cria regras de DQ em uma tabela baseada no Amazon S3 que se compara com o Amazon RDS ou. Amazon Redshift Nesses casos, o AWS Glue não consegue carregar a conexão. Em vez disso, tente configurar a regra de DQ no conjunto de dados Amazon Redshift ou no Amazon RDS. Esse é um erro conhecido.

ERRO DE INICIALIZAÇÃO: erro ao baixar o S3 para o bucket

Condição de erro: ERRO DE INICIALIZAÇÃO: erro ao baixar o S3 para o 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)` .

Resolução: As permissões na função passadas para a AWS Glue Data Quality devem permitir a leitura da localização anterior do Amazon S3. Essa política do IAM deve ser anexada ao perfil:

```
{
  "Sid": "allowS3",
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::aws-glue-ml-data-quality-assets-<region>/*"
}
```

Consulte a Autorização de qualidade de dados para ver as permissões detalhadas. Essas bibliotecas são necessárias para avaliar a qualidade dos dados de seus conjuntos de dados.

InvalidInputException (status: 400): DataQuality as regras não podem ser analisadas

Condição de erro: InvalidInputException (status: 400): DataQuality as regras não podem ser analisadas.

Resolução: existem muitas possibilidades para esse erro. Uma possibilidade é que suas regras tenham aspas simples. Verifique se elas estão entre aspas duplas. Por exemplo: .

```
Rules = [
ColumnValues "tipo_vinculo" in ["CODO", "DOCO", "COCO", "DODO"] AND "categoria" = 'ES"
    AND "cod_bandera" = 'CEP'
```

Altere isso para:

```
Rules = [
(ColumnValues "tipovinculo" in [ "CODO", "DOCO", "COCO", "DODO"]) AND (ColumnValues
 "categoria" = "ES")
```

```
    AND (ColumnValues "codbandera" = "CEP")
]
```

Erro: o Eventbridge não está acionando trabalhos do Glue DQ com base na agenda que eu configurei

Condição de erro: o Eventbridge não está acionando trabalhos do AWS Glue Data Quality com base na agenda que eu configurei.

Resolução: o perfil que aciona o trabalho pode não ter as permissões certas. Verifique se o perfil que você está usando para iniciar os trabalhos tem as permissões mencionadas em Configuração do IAM requerida para programar execuções de avaliação.

Erros de CustomSQL

Condição de erro: `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.`

Resolução: a consulta SQL é válida, mas verifique se as colunas selecionadas pertencem apenas à tabela primária. Selecionar funções agregadas como sum, count on the columns from the primary pode resultar nesse erro.

Condição de erro: `There was a problem when executing your SQL statement: cannot resolve "Col".`

Resolução: essa coluna não está presente na tabela primária.

Condição de erro: `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".`

Resolução: em consultas SQL, ao unir a tabela primária com outras tabelas de referência, verifique se a instrução de selecionar tem apenas nomes de colunas da tabela primária para gerar resultados em nível de linha para a tabela primária.

Regras dinâmicas

Condição de erro: `Dynamic rules require job context, and cannot be evaluated in interactive session or data preview.`.

Causa: essa mensagem de erro pode aparecer nos resultados da visualização de dados ou em outras sessões interativas quando as regras dinâmicas do DQ estiverem presentes no conjunto de regras. As regras dinâmicas fazem referência a métricas históricas associadas a um determinado nome de trabalho e contexto de avaliação. Portanto, não podem ser avaliadas em sessões interativas.

Resolução: a execução de trabalho do AWS Glue produzirá métricas históricas, que podem ser referenciadas em execuções posteriores do mesmo trabalho.

Condição de erro:

- `[RuleType] rule only supports simple atomic operands in thresholds.`.
- `Function last not yet implemented for [RuleType] rule.`

Resolução: as regras dinâmicas geralmente são compatíveis com todos os tipos de regras de DQDL em expressões numéricas (consulte Referência de DQDL). No entanto, algumas regras que produzem várias métricas ColumnValues e ColumnLength, ainda não são suportadas.

Condição de erro: `Binary expression operands must resolve to a single number.`.

Causa: as regras dinâmicas são compatíveis com expressões binárias, como `RowCount > avg(last(5)) * 0.9`. Nesse caso, a expressão binária é `avg(last(5)) * 0.9`. Essa regra é válida porque os dois operandos `avg(last(5))` e `0.9` se resolvem para um único número. `RowCount > last(5) * 0.9` é um exemplo incorreto, pois `last(5)` produzirá uma lista que não poderá ser comparada de forma significativa com a contagem de linhas atual.

Resolução: use funções de agregação para reduzir um operando com valores de lista para um único número.

Condição de erro:

- `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: as regras dinâmicas podem ser usadas para comparar alguns atributos do conjunto de dados com os valores históricos. A última função permite a recuperação de vários valores históricos, se um argumento inteiro positivo for fornecido. Por exemplo, `last(5)` recuperará os últimos cinco valores mais recentes observados nas execuções de trabalho da sua regra.

Resolução: uma função de agregação precisa ser usada para reduzir esses valores a um único número para fazer uma comparação significativa com o valor observado na execução do trabalho atual.

Exemplos válidos:

- `RowCount >= avg(last(5))`
- `RowCount > last(1)`
- `RowCount < last()`

Exemplo inválido: `RowCount > last(5)`.

Condição de erro:

- `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.`

Resolução: ao criar regras dinâmicas, você poderá usar a função de agregação `index` para selecionar um valor histórico em uma lista. Por exemplo, `RowCount > index(last(5), 1)` verificará se a contagem de linhas observada no trabalho atual é estritamente maior do que a segunda contagem de linhas mais recente observada no trabalho. `index` é indexado em zero.

Condição de erro: `IllegalArgumentException: Parsing Error: Rule Type: DetectAnomalies is not valid`.

Resolução: a detecção de anomalias só está disponível na versão 4.0 do AWS Glue.

Condição de erro: `IllegalArgumentException: Parsing Error: Unexpected condition for rule of type ... no viable alternative at input ...`.

Observação: `...` é dinâmico. Exemplo: `IllegalArgumentException: Parsing Error: Unexpected condition for rule of type RowCount with number return type, line 4:19 no viable alternative at input '>last'`.

Resolução: a detecção de anomalias só está disponível na versão 4.0 do AWS Glue.

Exceção na classe de usuário: org.apache.spark.sql. AnalysisException: org.apache.hadoop.hive.ql.metadata. HiveException

Condição de erro: `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: Você está usando o Apache Iceberg no AWS Glue Data Catalog e o atributo Input Format no AWS Glue Data Catalog está vazio.

Resolução: esse problema ocorre quando o tipo de regra CustomSQL é usado em sua regra de DQ. Uma maneira de corrigir isso é usar "primary" ou adicionar o nome do catálogo `glue_catalog.` a `<database>.<table> in Custom ruletype`.

UNCLASSIFIED_ERROR; IllegalArgumentException: Erro de análise: nenhuma regra ou analisador fornecido., nenhuma alternativa viável na entrada

Condição de erro: `UNCLASSIFIED_ERROR; IllegalArgumentException: Parsing Error: No rules or analyzers provided., no viable alternative at input`

Resolução: o DQDL não é analisável. Há alguns casos em que isso pode ocorrer. Se você estiver usando regras compostas, verifique se elas têm parênteses corretos.

```
(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
```

Integração de dados do Amazon Q em AWS Glue

A integração de dados do Amazon Q AWS Glue é um novo recurso generativo de IA AWS Glue que permite que engenheiros de dados e desenvolvedores de ETL criem trabalhos de integração de dados usando linguagem natural. Engenheiros e desenvolvedores podem pedir à Amazon Q que crie trabalhos, solucione problemas e responda perguntas sobre AWS Glue integração de dados.

O que é o Amazon Q?

> **Note**
>
> Desenvolvido pelo Amazon Bedrock: AWS implementa a detecção automática de abusos. Como a integração de dados do Amazon Q é baseada no Amazon Bedrock, os usuários podem aproveitar ao máximo os controles implementados no Amazon Bedrock para reforçar a segurança, a proteção e o uso responsável de inteligência artificial.

O Amazon Q é um assistente de conversação com inteligência artificial generativa (IA) que pode ajudar você a entender, criar, ampliar e operar AWS aplicativos. O modelo que impulsiona o Amazon Q foi aprimorado com AWS conteúdo de alta qualidade para oferecer respostas mais completas, acionáveis e referenciadas para acelerar sua construção. AWS Para obter mais informações, consulte What is Amazon Q?

O que é a integração de dados do Amazon Q no AWS Glue?

A integração de dados do Amazon Q AWS Glue inclui os seguintes recursos:

- Chat — A integração de dados do Amazon Q AWS Glue pode responder perguntas de linguagem natural em inglês sobre AWS Glue domínios de integração de dados, como conectores de AWS Glue origem e destino, tarefas de AWS Glue ETL, catálogo de dados, rastreadores e AWS Lake Formation outras documentações de recursos e melhores práticas. A integração de dados do Amazon Q AWS Glue responde com step-by-step instruções e inclui referências às suas fontes de informação.
- Geração de código de integração de dados — A integração de dados do Amazon Q AWS Glue pode responder perguntas sobre scripts de AWS Glue ETL e gerar novos códigos com base em uma pergunta de linguagem natural em inglês.

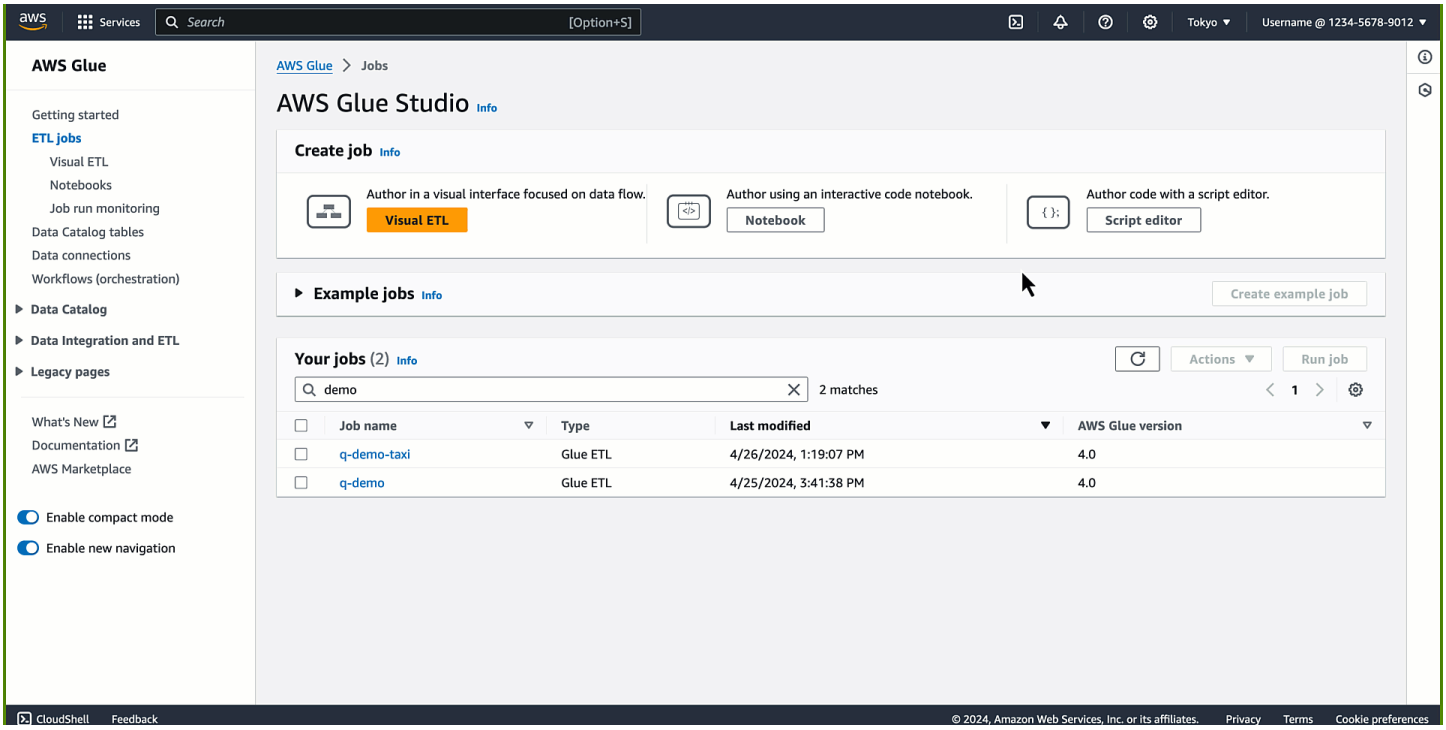
- Solução de problemas — A integração de dados do Amazon Q foi criada especificamente para ajudar você a entender erros em AWS Glue trabalhos e fornece step-by-step instruções para identificar a causa e resolver seus problemas. AWS Glue

> **Note**
>
> A integração de dados do Amazon Q AWS Glue não usa o contexto da sua conversa para informar futuras respostas durante a conversa. Cada conversa com a integração de dados do Amazon Q AWS Glue é independente de suas conversas anteriores ou futuras.

Como trabalhar com a integração de dados do Amazon Q no AWS Glue?

No painel do Amazon Q, você pode solicitar que o Amazon Q gere código para um script de AWS Glue ETL ou responda a uma pergunta sobre AWS Glue recursos ou solucione um erro. A resposta é um script ETL PySpark com step-by-step instruções para personalizar o script, revisá-lo e executá-lo. Para perguntas, a resposta é gerada de acordo com a base de conhecimento de integração de dados com um resumo e um URL de origem para referências.

Por exemplo, você pode pedir ao Amazon Q que “Forneça um script Glue que leia do Snowflake, renomeie os campos e grave no Redshift” e, em resposta, a integração de dados do Amazon Q AWS Glue retornará um script de AWS Glue trabalho que possa realizar a ação solicitada. Você pode revisar o código gerado para garantir que ele atenda à intenção solicitada. Se estiver satisfeito, você pode implantá-lo como um AWS Glue trabalho em produção. Você pode solucionar problemas solicitando que a integração explique erros e falhas e proponha soluções. O Amazon Q pode responder perguntas sobre AWS Glue nossas melhores práticas de integração de dados.



A seguir estão exemplos de perguntas que demonstram como a integração de dados do Amazon Q AWS Glue pode ajudar você a desenvolver AWS Glue:

AWS Glue Geração de código ETL:

- Escreva um AWS Glue script que leia o JSON do S3, transforme campos usando o mapeamento de aplicação e grave no Amazon Redshift
- Como escrevo um AWS Glue script para ler no DynamoDB, aplicar a transformação e gravar DropNullFields no S3 como Parquet?
- Forneça um AWS Glue script que leia o MySQL, elimine alguns campos com base na minha lógica de negócios e grave no Snowflake
- Escrever um AWS Glue trabalho para ler do DynamoDB e gravar no S3 como JSON
- Ajude-me a desenvolver um AWS Glue script para o AWS Glue Data Catalog to S3
- Escreva um AWS Glue trabalho para ler JSON do S3, eliminar nulos e gravar no Redshift

AWS Glue explicações de características:

- Como faço para usar a qualidade de AWS Glue dados?
- Como usar marcadores AWS Glue de emprego?
- Como faço para ativar o AWS Glue escalonamento automático?

- Qual é a diferença entre quadros AWS Glue dinâmicos e quadros de dados do Spark?
- Quais são os diferentes tipos de conexões compatíveis AWS Glue?

AWS Glue solução de problemas:

- Como solucionar erros de falta de memória (OOM) em AWS Glue trabalhos?
- Quais são algumas mensagens de erro que você pode ver ao configurar a Qualidade de AWS Glue Dados e como corrigi-las?
- Como faço para corrigir um AWS Glue trabalho com o erro de acesso negado ao Amazon S3?
- Como faço para resolver problemas com o embaralhamento de dados em AWS Glue trabalhos?

Melhores práticas para interagir com a integração de dados do Amazon Q

A seguir estão as melhores práticas para interagir com a integração de dados do Amazon Q:

- Ao interagir com a integração de dados do Amazon Q, faça perguntas específicas, repita quando tiver solicitações complexas e verifique se as respostas estão corretas.
- Ao fornecer instruções de integração de dados em linguagem natural, seja o mais específico possível para ajudar o assistente a entender exatamente o que você precisa. Em vez de pedir “extrair dados do S3”, forneça mais detalhes, como “escrever um AWS Glue script que extraia arquivos JSON do S3”.
- Revise o script gerado antes de executá-lo para garantir a precisão. Se o script gerado tiver erros ou não corresponder à sua intenção, forneça instruções ao assistente sobre como corrigi-lo.
- A tecnologia de IA generativa é nova e pode haver erros, às vezes chamados de alucinações, nas respostas. Teste e analise todo o código em busca de erros e vulnerabilidades antes de usá-lo no ambiente ou no workload.

Integração de dados do Amazon Q na melhoria AWS Glue de serviços

Para ajudar a integração de dados do Amazon Q a AWS Glue fornecer as informações mais relevantes sobre AWS os serviços, podemos usar determinados conteúdos do Amazon Q, como perguntas que você faz ao Amazon Q e suas respostas, para melhorar o serviço.

Para obter informações sobre o conteúdo que usamos e como optar por não participar, consulte Melhoria do serviço Amazon Q Developer no Amazon Q Developer User Guide.

Considerações

Considere os seguintes itens antes de usar a integração de dados do Amazon Q no AWS Glue:

- Atualmente, a geração de código só funciona com o PySpark kernel. O código gerado é para AWS Glue trabalhos baseados no Python Spark.
- Para obter informações sobre as combinações suportadas de habilidades de geração de código da integração de dados do Amazon Q em AWS Glue, consulteCapacidades de geração de código suportadas.

Configurando a integração de dados do Amazon Q em AWS Glue

As seções a seguir fornecem informações sobre como configurar a integração de dados do Amazon Q no AWS Glue.

Tópicos

- Configurar permissões do IAM

Configurar permissões do IAM

Este tópico descreve as permissões do IAM que você configura para a experiência de bate-papo do Amazon Q e a experiência do notebook AWS Glue Studio.

Tópicos

- Configurando permissões do IAM para o Amazon Q chat
- Configurando permissões do IAM para notebooks AWS Glue Studio

Configurando permissões do IAM para o Amazon Q chat

A concessão de permissões às APIs usadas pela integração de dados do Amazon Q AWS Glue exige permissões apropriadas de AWS Identity and Access Management (IAM). Você pode obter permissões anexando a seguinte AWS política personalizada à sua identidade do IAM (como usuário, função ou grupo):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartCompletion",
                "glue:GetCompletion"
            ],
            "Resource": [
                "arn:aws:glue:*:*:completion/*"
            ]
        }
    ]
}
```

Configurando permissões do IAM para notebooks AWS Glue Studio

Para habilitar a integração de dados do Amazon Q em notebooks AWS Glue Studio, certifique-se de que a seguinte permissão esteja anexada à função do IAM do notebook:

```
{
    "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**
>
> A integração de dados do Amazon Q AWS Glue não tem APIs disponíveis por meio do AWS SDK que você possa usar programaticamente. As duas APIs a seguir são usadas na política do IAM para permitir essa experiência por meio do painel de bate-papo do Amazon Q ou dos notebooks AWS Glue Studio: e. `StartCompletion` `GetCompletion`

Atribuindo permissões

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos no AWS IAM Identity Center: crie um conjunto de permissões. Siga as instruções em Criar um conjunto de permissões no Guia do usuário do Centro de Identidade do AWS IAM.
- Usuários gerenciados no IAM por meio de um provedor de identidades: criem um perfil para federação de identidades. Siga as instruções em Criar um perfil para um provedor de identidades de terceiros (federação) no Guia do usuário do IAM.
- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

Capacidades de geração de código suportadas

A seguir estão as combinações das capacidades de geração de código da integração de dados do Amazon Q no AWS Glue.

Origem	Transformação	Destino
S3 com os seguintes tipos de formato: json, csv, parquet, hudi, delta	ApplyMapping	S3 com os seguintes tipos de formato: json, csv, avro, orc, parquet, hudi, delta
Glue Data Catalog	RenameField	Glue Data Catalog
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	Agregar	Snowflake
MongoDB	DropDuplicates	MongoDB
Conector JDBC personalizado	Ingressar	Conector JDBC personalizado
Conector Spark personalizado	Union	Conector Spark personalizado
Google BigQuery		Google BigQuery
Teradata		Teradata
OpenSearch Serviço Amazon		OpenSearch Serviço Amazon
Vertica		Vertica
SQL do Azure		SQL do Azure
SAP HANA		SAP HANA
Cosmos Azure		Cosmos Azure

Interações de exemplo

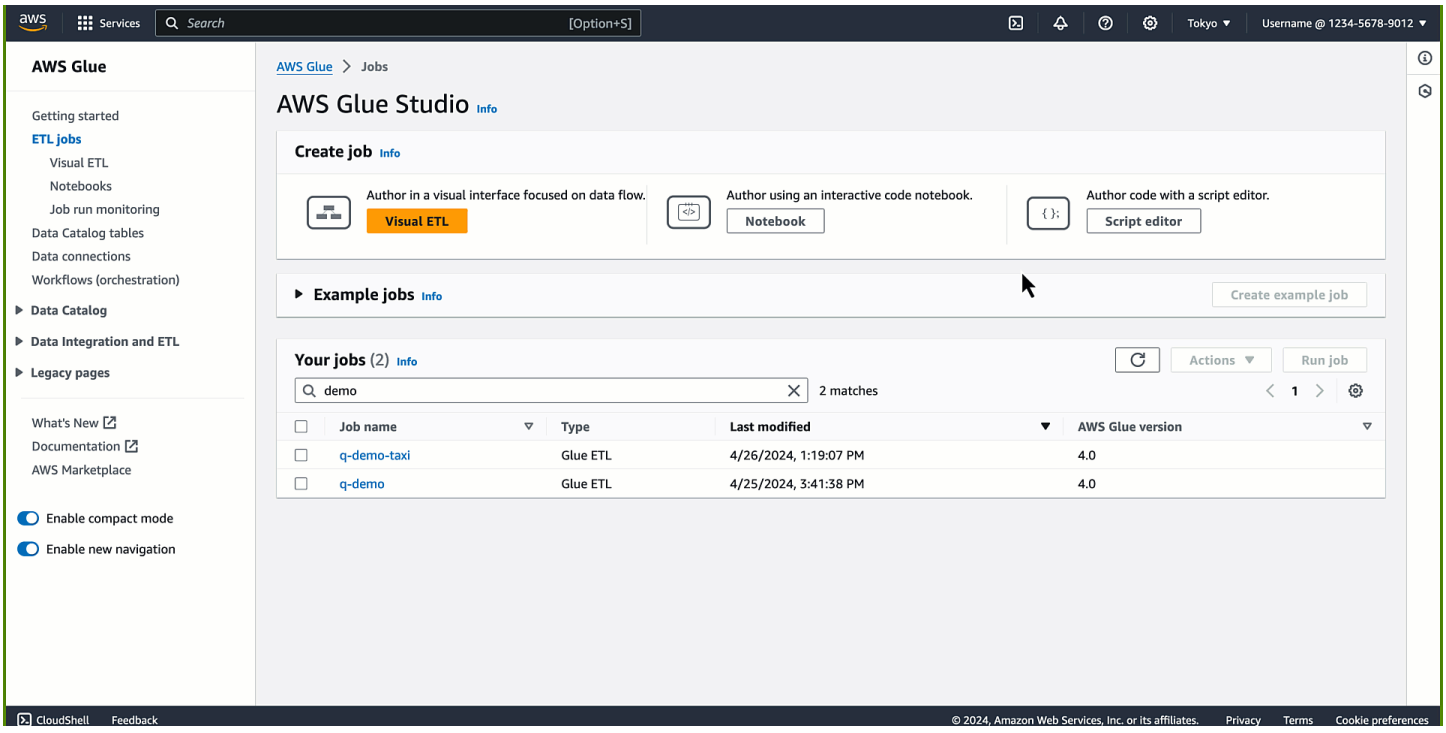
A integração de dados do Amazon Q AWS Glue permite que você insira sua pergunta no painel do Amazon Q. Você pode inserir uma pergunta sobre a funcionalidade de integração de dados fornecida pelo AWS Glue. Uma resposta detalhada, junto com documentos de referência, será retornada.

Outro caso de uso é gerar scripts de tarefas AWS Glue ETL. Você pode fazer uma pergunta sobre como realizar um trabalho de extração, transformação e carregamento de dados. Um PySpark script gerado será retornado.

Tópicos

Interações de chat do Amazon Q

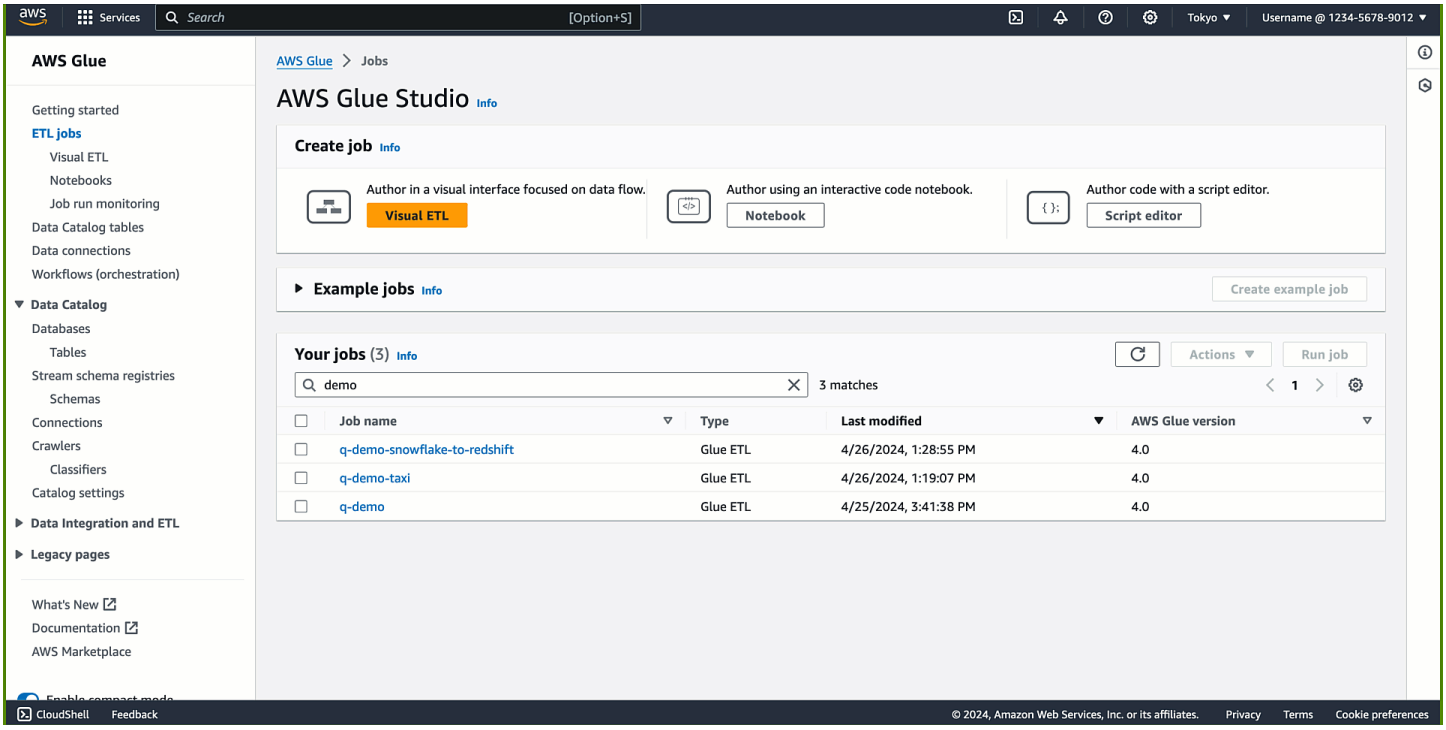
No AWS Glue console, comece a criar um novo trabalho e pergunte à Amazon Q: “Forneça um script Glue que leia do Snowflake, renomeie os campos e grave no Redshift”.



Você notará que o código foi gerado. Com essa resposta, você pode aprender e entender como criar AWS Glue código para sua finalidade. Você pode copiar/colar o código gerado no editor de scripts

e configurar espaços reservados. Depois de configurar uma função AWS Glue e conexões do AWS Identity and Access Management (IAM) no trabalho, salve e execute o trabalho. Quando o trabalho estiver concluído, você poderá começar a consultar a tabela exportada do Snowflake no Amazon Redshift.

O prompt a seguir lê dados de duas fontes diferentes, os filtra e os projeta individualmente, une em uma chave comum e grava a saída em um terceiro destino. Pergunte à Amazon Q: “Quero ler dados do S3 no formato Parquet e selecionar alguns campos. Também quero ler dados do DynamoDB, selecionar alguns campos e filtrar algumas linhas. Quero unir esses dois conjuntos de dados e gravar os resultados em. OpenSearch

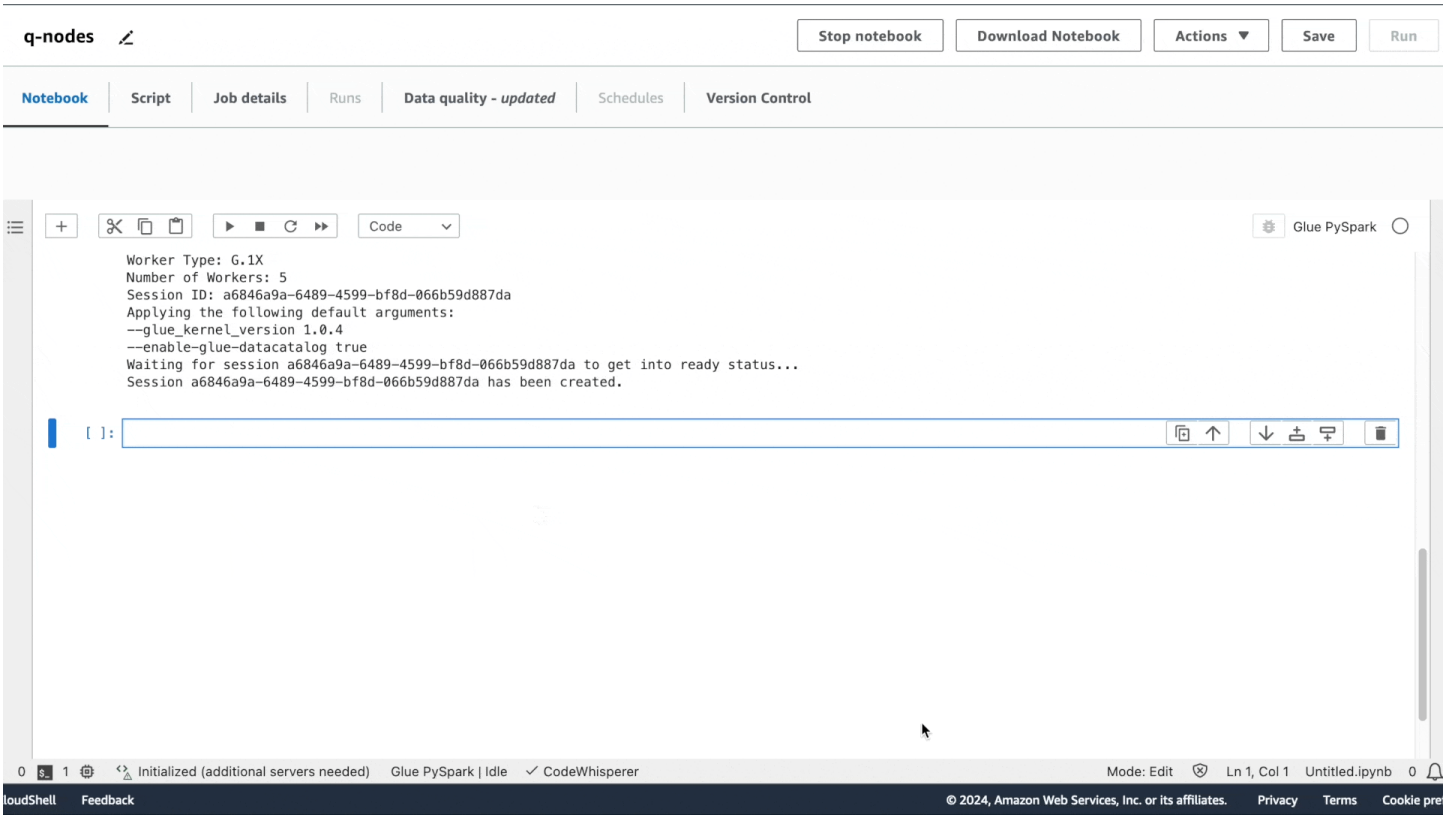


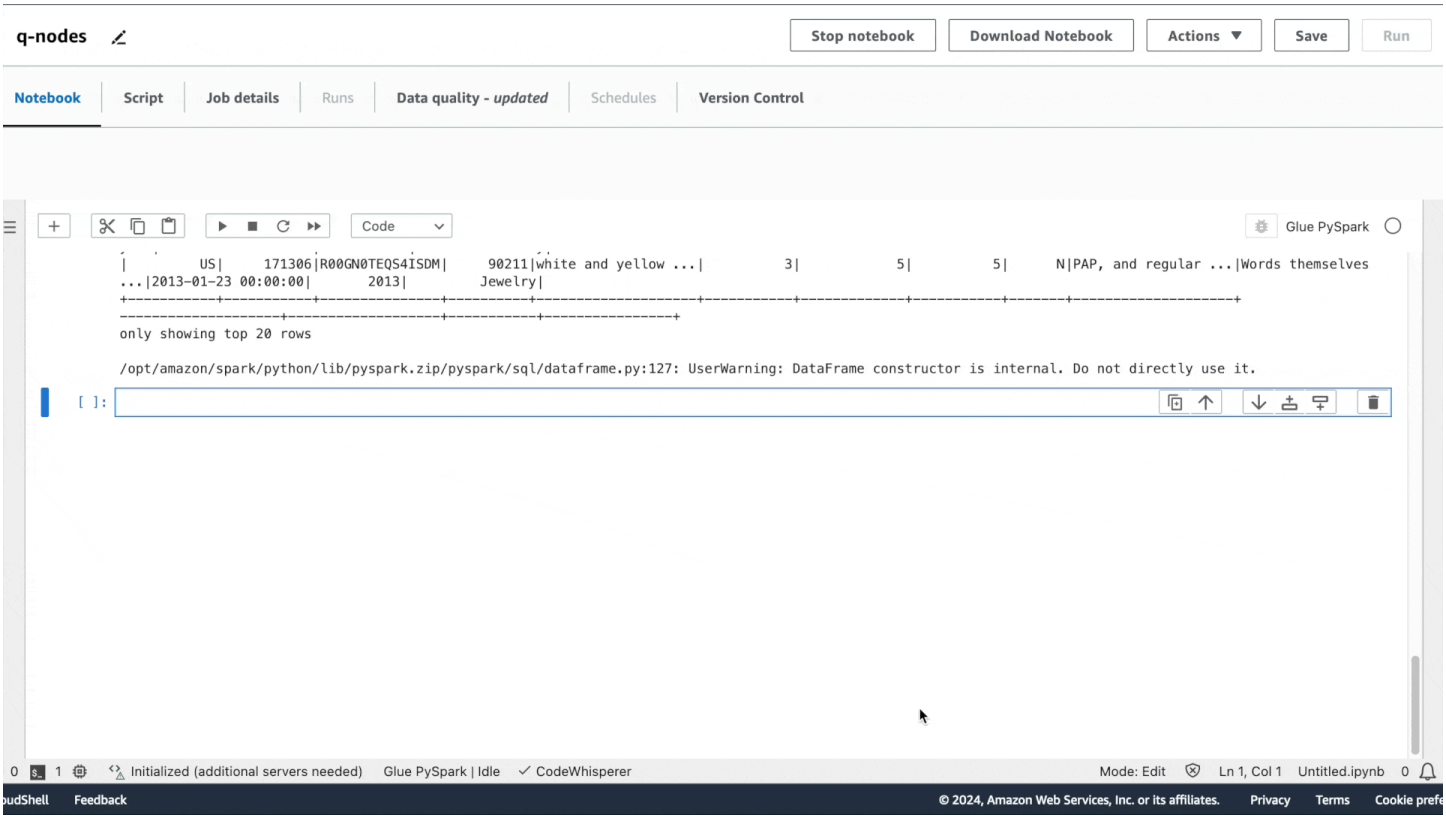
O código é gerado. Quando o trabalho estiver concluído, seu índice estará disponível OpenSearch e poderá ser usado por suas cargas de trabalho posteriores.

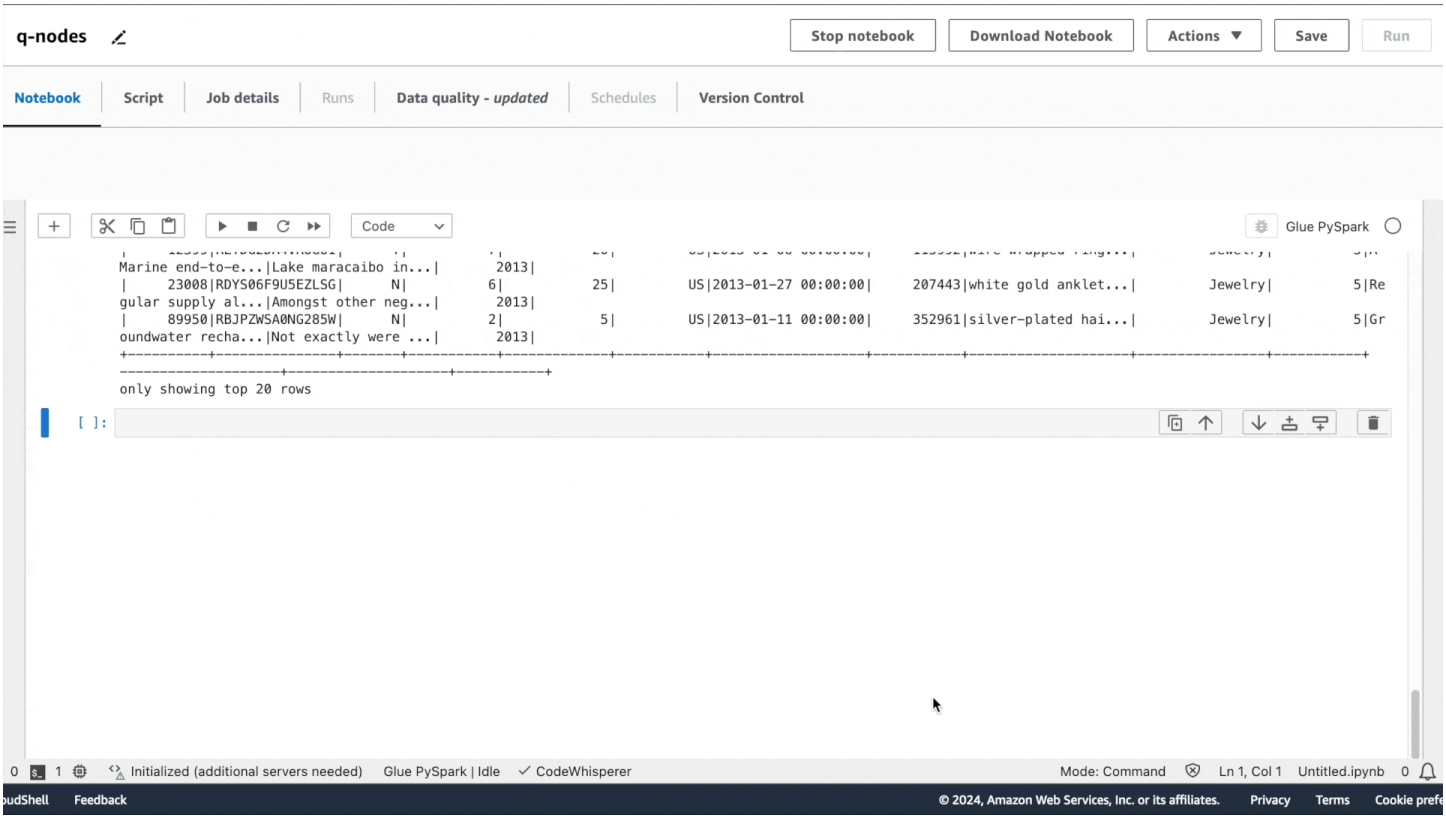
AWS Glue Interações com notebooks de estúdio

Adicione uma nova célula e insira seu comentário para descrever o que você deseja alcançar. Depois de pressionar Tab e Enter, o código recomendado é exibido.

A primeira intenção é extrair os dados: “Forneça um código que leia uma tabela do Glue Data Catalog”, seguido por “Forneça um código para aplicar uma transformação de filtro com star_rating>3" e “Forneça o código que grava o quadro no 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

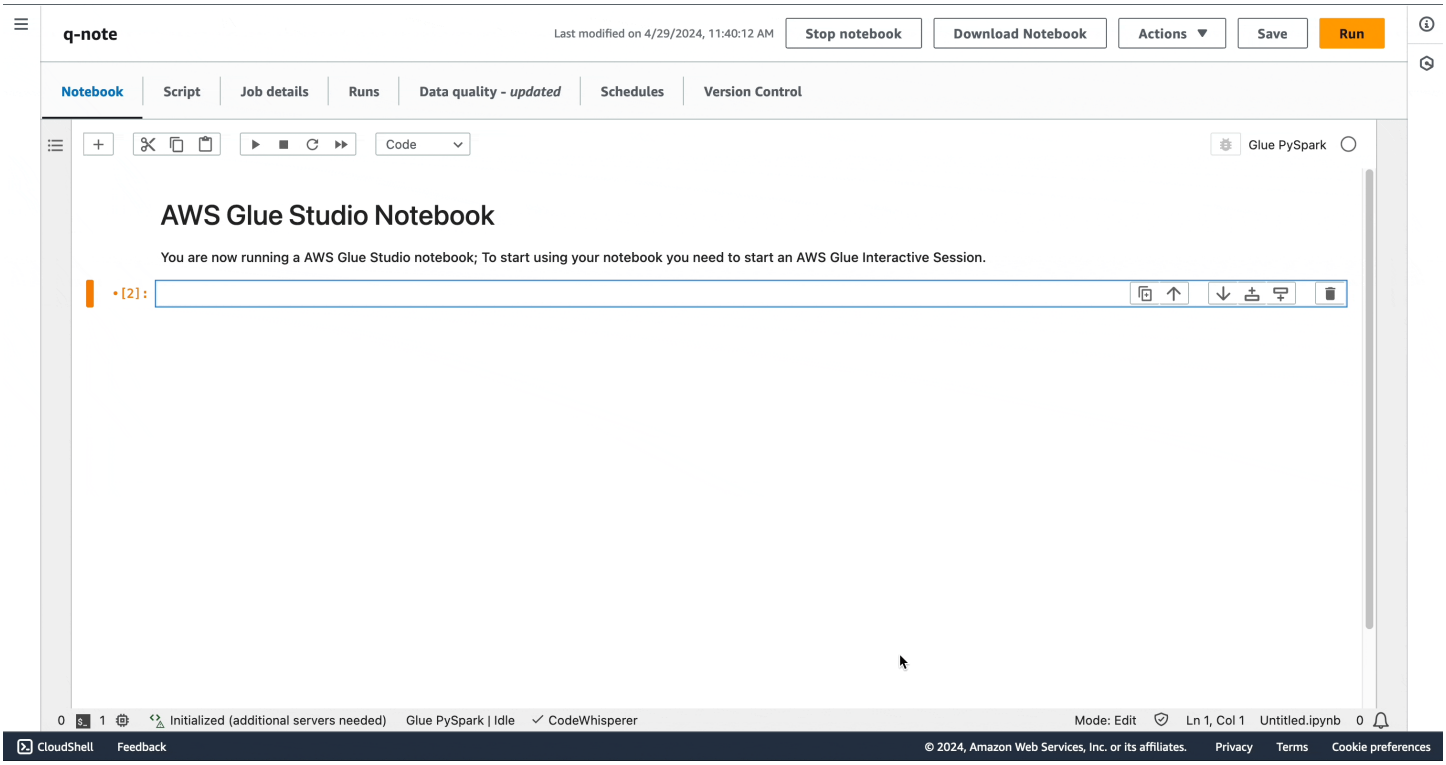


Semelhante à experiência de bate-papo do Amazon Q, o código é recomendado. Se você pressionar Tab, o código recomendado será escolhido.

Você pode executar cada célula preenchendo as opções apropriadas para suas fontes no código gerado. Em qualquer momento das execuções, você também pode visualizar uma amostra do seu conjunto de dados usando o `show( )` método.

Solicitações complexas

Você pode gerar um script completo com um único prompt complexo. “Tenho dados JSON no S3 e dados no Oracle que precisam ser combinados. Forneça um script Glue que leia as duas fontes, faça uma junção e, em seguida, grave os resultados no Redshift.”



Você pode notar que, no notebook, a integração de dados do Amazon Q AWS Glue gerou o mesmo trecho de código que foi gerado no chat do Amazon Q.

Você pode executar o notebook como um trabalho, escolhendo Executar ou programaticamente.

Orquestração no AWS Glue

As seções a seguir fornecem mais informações sobre a orquestração de trabalhos no AWS Glue.

Tópicos

Iniciar trabalhos e crawlers usando acionadores

No AWS Glue, você pode criar objetos do Data Catalog chamados acionadores, que podem ser usados para iniciar manual ou automaticamente um ou mais crawlers, ou extrair, transformar e carregar (ETL) trabalhos. Ao usar gatilhos, você pode projetar uma cadeia de trabalhos e crawlers dependentes.

> **Note**
>
> É possível fazer a mesma coisa definindo fluxos de trabalho. Os fluxos de trabalho são preferidos para a criação de operações ETL complexas com vários trabalhos. Para ter mais informações, consulte the section called “Realizar atividades de ETL complexas usando esquemas e fluxos de trabalho”.

Tópicos

Gatilhos do AWS Glue

Quando acionado, um gatilho pode iniciar trabalhos e crawlers especificados. Um gatilho é acionado sob demanda, com base em uma programação ou em uma combinação de eventos.

> **Note**
>
> Apenas dois crawlers podem ser ativados por um único acionador. Se você quiser realizar crawling em vários armazenamentos de dados, use várias fontes para cada crawler, em vez de executar vários crawlers simultaneamente.

Um gatilho pode existir em um de vários estados. Um gatilho é `CREATED`, `ACTIVATED` ou `DEACTIVATED`. Existem também estados transitórios, como `ACTIVATING`. Para impedir temporariamente que um gatilho seja acionado, é possível desabilitá-lo. Você pode reabilitá-lo posteriormente.

Existem três tipos de gatilhos:

Programado

Um gatilho de tempo baseado em `cron`.

Você pode criar um gatilho para um conjunto de trabalhos ou crawlers com base em uma programação. Você pode especificar restrições, como a frequência de execução ou em quais dias da semana e a que horas os trabalhos ou crawlers são executados. Essas restrições são feitas com base no `cron`. Ao configurar a programação de um gatilho, considere os recursos e as limitações do cron. Por exemplo, se você optar por executar seu crawler no dia 31 de cada mês, lembre-se de que alguns meses não têm 31 dias. Para obter mais informações sobre o cron, consulte Programações baseadas em hora para trabalhos e crawlers.

Condicional

Um gatilho que é acionado quando um ou vários trabalhos ou crawlers satisfazem uma lista de condições.

Ao criar um gatilho condicional, você especifica uma lista de trabalhos e uma lista de crawlers a serem monitorados. Para cada trabalho ou crawler monitorado, você especifica um status a monitorar, como succeeded (bem-sucedido), failed (com falha), timed out (com tempo limite excedido) e assim por diante. O gatilho será acionado se os trabalhos ou crawlers monitorados forem finalizados com os status especificados. Você pode configurar o gatilho de modo a ser acionado quando um ou todos os eventos monitorados ocorrerem.

Por exemplo, você pode configurar um gatilho T1 para iniciar o trabalho J3 quando o trabalho J1 e J2 forem concluídos com êxito, e outro gatilho T2 para iniciar o trabalho J4 se houver falha no trabalho J1 ou no J2.

A tabela a seguir lista os estados de conclusão do trabalho e do crawler (eventos) que acionam a gatilho.

Estados de conclusão do trabalho	Estados de conclusão do crawler
• `SUCCEEDED` • `STOPPED` • `FAILED` • `TIMEOUT`	• `SUCCEEDED` • `FAILED` • `CANCELLED`

Sob demanda

Um gatilho que é acionado quando é ativado. Os gatilhos sob demanda nunca entram no estado `ACTIVATED` ou `DEACTIVATED`. Eles sempre permanecem no estado `CREATED`.

Para que eles estejam prontos para serem acionados assim que existirem, você pode definir um sinalizador para ativar gatilhos programados e condicionais ao criá-los.

> **Important**
>
> Os trabalhos ou crawlers executados como resultado da conclusão de outros trabalhos ou crawlers são chamados de dependentes. Os trabalhos ou crawlers dependentes só serão iniciados se o trabalho ou o crawler concluído tiver sido iniciado por um gatilho. Todos os trabalhos ou crawlers em uma cadeia de dependência devem ser descendentes de um único gatilho scheduled (programado) ou on-demand (sob demanda).

Transmitir parâmetros de trabalho com acionadores

Um gatilho pode transmitir parâmetros para os trabalhos que inicia. Os parâmetros incluem argumentos de trabalho, valor de tempo limite, configuração de segurança e muito mais. Se o gatilho iniciar vários trabalhos, os parâmetros serão transmitidos para cada trabalho.

Veja a seguir as regras para argumentos de trabalho transmitidos por um gatilho:

- Se a chave no par de chave/valor corresponder a um argumento de trabalho padrão, o argumento transmitido substituirá o argumento padrão. Se a chave não corresponder a um argumento padrão, o argumento será transmitido como um argumento adicional para o trabalho.
- Se a chave no par de chave/valor corresponder a um argumento não substituível, o argumento transmitido será ignorado.

Para obter mais informações, consulte the section called “Acionadores” na API do AWS Glue.

Adição de acionadores

É possível adicionar um gatilho usando o console do AWS Glue, a AWS Command Line Interface (AWS CLI) ou a API do AWS Glue.

> **Note**
>
> Atualmente, o console do AWS Glue oferece suporte somente a trabalhos, não a crawlers, ao trabalhar com gatilhos. Você pode usar a AWS CLI ou a API do AWS Glue para configurar gatilhos com trabalhos e crawlers.

Como adicionar um gatilho (console)

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em ETL, selecione Triggers (Gatilhos). Escolha Add trigger (Adicionar gatilhos).
3. Forneça as seguintes propriedades:

   Nome

   Atribua um nome exclusivo ao gatilho.

   Tipos de gatilho

   Especifique um dos seguintes:

   - Schedule (Programação): o gatilho é acionado em uma frequência e hora específicas.
   - Job events (Eventos de trabalho): um gatilho condicional. O gatilho é acionado quando um ou todos os trabalhos na lista correspondem ao seu status designado. Para que o gatilho

seja acionado, os trabalhos monitorados devem ter sido iniciados por um gatilho. Para qualquer trabalho que você escolher, só é possível observar um evento de trabalho (status de conclusão).

- On-demand (Sob demanda): o gatilho é acionado quando habilitado.

4. Executar o assistente de gatilho. Na página Review (Revisar) você pode ativar gatilhos Schedule (Programados) e de Job events (Eventos de trabalho) (condicionais) imediatamente selecionando Enable trigger on creation (Habilitar gatilho na criação).

Como adicionar um gatilho (AWS CLI)

- Digite um comando semelhante ao seguinte:

```
aws glue create-trigger --name MyTrigger --type SCHEDULED --schedule  "cron(0 12 *
 * ? *)" --actions CrawlerName=MyCrawler --start-on-creation
```

Esse comando cria um gatilho programado chamado `MyTrigger`, que é executado todos os dias às 12h UTC e inicia um crawler chamado `MyCrawler`. O gatilho é criado no estado ativado.

Para ter mais informações, consulte the section called “Gatilhos do AWS Glue”.

Programações baseadas em hora para trabalhos e crawlers

Você pode definir uma programação baseada em hora para seus crawlers e trabalhos no AWS Glue. A definição dessas programações usa a sintaxe cron semelhante à do Unix. Você especifica a hora em Tempo Universal Coordenado (UTC). Além disso, a precisão mínima para uma programação é de 5 minutos.

Para saber mais sobre como configurar trabalhos e crawlers para serem executados usando uma programação, consulte Iniciar trabalhos e crawlers usando acionadores.

Expressão cron

Expressões cron têm seis campos obrigatórios, que são separados por um espaço em branco.

Sintaxe

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

Campos	Valores	Curingas
Minutos	0–59	, - * /
Horas	0–23	, - * /
Dia do mês	1–31	, - * ? / L W
Mês	1-12 ou JAN-DEZ	, - * /
Dia da semana	1-7 ou SUN-SAT	, - * ? / L
Ano	1970–2199	, - * /

Curingas

- A , (vírgula) curinga inclui valores adicionais. No campo `Month`, `JAN,FEB,MAR` incluirá janeiro, fevereiro e março.
- O - (traço) curinga especifica intervalos. No campo `Day`, 1-15 incluiria os dias 1 a 15 do mês especificado.
- O * (asterisco) curinga inclui todos os valores no campo. No campo `Hours`, * incluirá cada hora.
- A / (barra) curinga especifica incrementos. No campo `Minutes`, você pode inserir **`1/10`** para especificar cada décimo minuto a partir do primeiro minuto da hora (por exemplo, o 11º, 21º e 31º minuto, etc).
- O curinga ? (interrogação) especifica um ou outro. No campo `Day-of-month`, você pode inserir `7` e, se não se importar com qual dia da semana era o sétimo, pode inserir ? no campo Dia da semana.
- O curinga L nos campos `Day-of-month` ou `Day-of-week` especifica o último dia do mês ou da semana.
- O curinga W no campo `Day-of-month` especifica um dia da semana. No campo `Day-of-month`, `3W` especifica o dia mais próximo do terceiro dia da semana do mês.

Limites

- Não é possível especificar os campos `Day-of-month` e `Day-of-week` na mesma expressão cron. Se você especificar um valor em um dos campos, deverá usar um ? (ponto de interrogação) no outro.
- As expressões cron que levam a taxas mais rápidas do que 5 minutos não têm suporte.

Exemplos

Ao criar uma programação, você pode usar os seguintes exemplos de strings cron.

Minutos	Horas	Dia do mês	Mês	Dia da semana	Ano	Significado
0	10	*	*	?	*	Executada às 10h (UTC) todos os dias
15	12	*	*	?	*	Executada às 12h15 (UTC) todos os dias
0	18	?	*	SEG-SEX	*	Executada às 18h (UTC) de segunda a sexta
0	8	1	*	?	*	Executada às 8h (UTC) todo o primeiro dia do mês

Minutos	Horas	Dia do mês	Mês	Dia da semana	Ano	Significado
0/15	*	*	*	?	*	Executada a cada 15 minutos
0/10	*	?	*	SEG-SEX	*	Executada a cada 10 minutos de segunda a sexta
0/5	8-17	?	*	SEG-SEX	*	Executada a cada 5 minutos de segunda a sexta entre 8h e 17h55 (UTC)

Por exemplo, para executar uma programação todos os dias às 12h15 UTC, especifique:

```
cron(15 12 * * ? *)
```

Ativar e desativar acionadores

Você pode ativar ou desativar um gatilho usando o AWS Glue console, o AWS Command Line Interface (AWS CLI) ou a AWS Glue API.

Como ativar ou desativar um gatilho (console)

1. Faça login no AWS Management Console e abra o AWS Glue console em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em ETL, selecione Triggers (Gatilhos).

3. Marque a caixa de seleção ao lado do gatilho desejado e, no menu Action (Ação), escolha Enable trigger (Habilitar gatilho) para ativar o gatilho ou Disable trigger (Desabilitar gatilho) para desativar o gatilho.

Como ativar ou desativar um gatilho (AWS CLI)

- Insira um dos seguintes comandos:

```
aws glue start-trigger --name MyTrigger

aws glue stop-trigger --name MyTrigger
```

  Ao iniciar um gatilho ele é ativado e ao interromper um gatilho ele é desativado. Quando você ativa um gatilho sob demanda, ele é acionado imediatamente.

Para ter mais informações, consulte the section called “Gatilhos do AWS Glue”.

Realizar atividades de ETL complexas usando esquemas e fluxos de trabalho no AWS Glue

Alguns dos processos complexos de extração, transformação e carregamento (ETL) da sua organização podem ser melhor implantados usando vários crawlers e trabalhos dependentes do AWS Glue. Usando fluxos de trabalho do AWS Glue, você pode projetar um processo complexo de ETL com vários trabalhos e crawlers, que o AWS Glue pode executar e rastrear como uma entidade única. Depois de criar um fluxo de trabalho e especificar os trabalhos, crawlers e acionadores no fluxo de trabalho, você pode executá-lo sob demanda ou em uma programação.

Tópicos

- Obter e configurar as propriedades de execução de fluxo de trabalho no AWS Glue
- Consultar fluxos de trabalho usando o AWS Glue API
- Restrições de esquema e fluxo de trabalho no AWS Glue
- Solucionar erros de esquema no AWS Glue
- Permissões para pessoas e funções de esquemas do AWS Glue

Visão geral de fluxos de trabalho no AWS Glue

No AWS Glue, você pode usar fluxos de trabalho para criar e visualizar atividades complexas de extração, transformação e carregamento (ETL) que envolvem vários crawlers, trabalhos e gatilhos. Cada fluxo de trabalho gerencia a execução e o monitoramento de todos os seus trabalhos e crawlers. À medida que um fluxo de trabalho executa cada componente, ele registra o progresso e o status de execução. Isso fornece a você uma visão geral das tarefa maior e os detalhes de cada etapa. O console do AWS Glue fornece uma representação visual de um fluxo de trabalho como um gráfico.

Você pode criar um fluxo de trabalho a partir de um blueprint do AWS Glue ou pode construir um fluxo de trabalho manualmente, um componente de cada vez, usando o AWS Management Console ou a AWS Glue API. Para obter mais informações sobre esquemas, consulte the section called “Visão geral dos esquemas”.

Os acionadores nos fluxos de trabalho podem ser acionar trabalhos e crawlers e podem ser disparados quando trabalhos e crawlers são concluídos. Usando acionadores, é possível criar grandes cadeias de trabalhos e crawlers interdependentes. Além dos acionadores em um fluxo de trabalho que definem dependências de trabalhos e crawlers, cada fluxo de trabalho tem um acionador de início. Existem três tipos de acionadores de início:

- Schedule (Programação): o fluxo de trabalho é iniciado de acordo com a programação definida por você. A programação pode ser diária, semanal, mensal e assim por diante, ou pode ser uma programação personalizada com base em uma expressão `cron`.
- On demand (Sob demanda): o fluxo de trabalho é iniciado manualmente pelo console do AWS Glue, API ou AWS CLI.
- EventBridge event (Evento do EventBridge): o fluxo de trabalho é iniciado após a ocorrência de um único evento do Amazon EventBridge ou de um lote de eventos do Amazon EventBridge. Com esse tipo de acionador, o AWS Glue pode ser um consumidor de eventos em uma arquitetura orientada por eventos. Qualquer evento do EventBridge pode iniciar um fluxo de trabalho. O mais

provável caso de uso é a chegada de um novo objeto em um bucket do Amazon S3 (a operação `PutObject` do S3).

Iniciar um fluxo de trabalho com um lote de eventos significa esperar até que um número especificado de eventos tenha sido recebido ou até que um determinado período de tempo tenha passado. Ao criar o acionador de evento do EventBridge, você pode, opcionalmente, especificar condições de lote. Se você especificar condições de lote, deverá especificar o tamanho do lote (número de eventos) e, opcionalmente, poderá especificar uma janela de lote (número de segundos). A janela padrão e máxima do lote é 900 segundos (15 minutos). A condição de lote que é atendida primeiro inicia o fluxo de trabalho. A janela de lote é iniciada quando o primeiro evento chega. Se você não especificar condições de lote ao criar um acionador, o tamanho do lote assumirá como padrão 1.

Quando o fluxo de trabalho é iniciado, as condições de lote são redefinidas e o acionador de evento começa a observar a próxima condição de lote a ser atendida para iniciar o fluxo de trabalho novamente.

A tabela a seguir mostra como o tamanho do lote e a janela do lote operam juntos para acionar um fluxo de trabalho.

Tamanho do lote	Janela do lote	Condição de acionamento resultante
10		O fluxo de trabalho é acionado após a chegada de dez eventos do EventBridge ou 15 minutos após a chegada do primeiro evento, o que ocorrer primeiro. (se o tamanho das janelas não for especificado, o padrão será 15 minutos).
10	Dois minutos	O fluxo de trabalho é acionado após a chegada de dez eventos do EventBridge ou dois minutos após a chegada do primeiro evento, o que ocorrer primeiro.
1		O fluxo de trabalho é acionado após a chegada do primeiro evento. O tamanho da janela é irrelevante. Se você não especificar condições de lote ao criar um acionador de evento do EventBridge, o tamanho do lote assumirá como padrão 1.

A operação de API `GetWorkflowRun` retorna a condição de lote que acionou o fluxo de trabalho.

Independentemente de como um fluxo de trabalho é iniciado, você pode especificar o número máximo de execuções de fluxo de trabalho simultâneas ao criar o fluxo de trabalho.

Se um evento ou lote de eventos iniciar uma execução de fluxo de trabalho que falha em algum momento, esse evento ou lote de eventos não será mais considerado para iniciar uma execução de fluxo de trabalho. Uma nova execução de fluxo de trabalho é iniciada somente quando o próximo evento ou lote de eventos chega.

> **Important**
>
> Limite o número total de trabalhos, crawlers e acionadores em um fluxo de trabalho a 100 ou menos. Se você incluir mais de 100, poderá receber erros ao tentar retomar ou interromper as execuções do fluxo de trabalho.

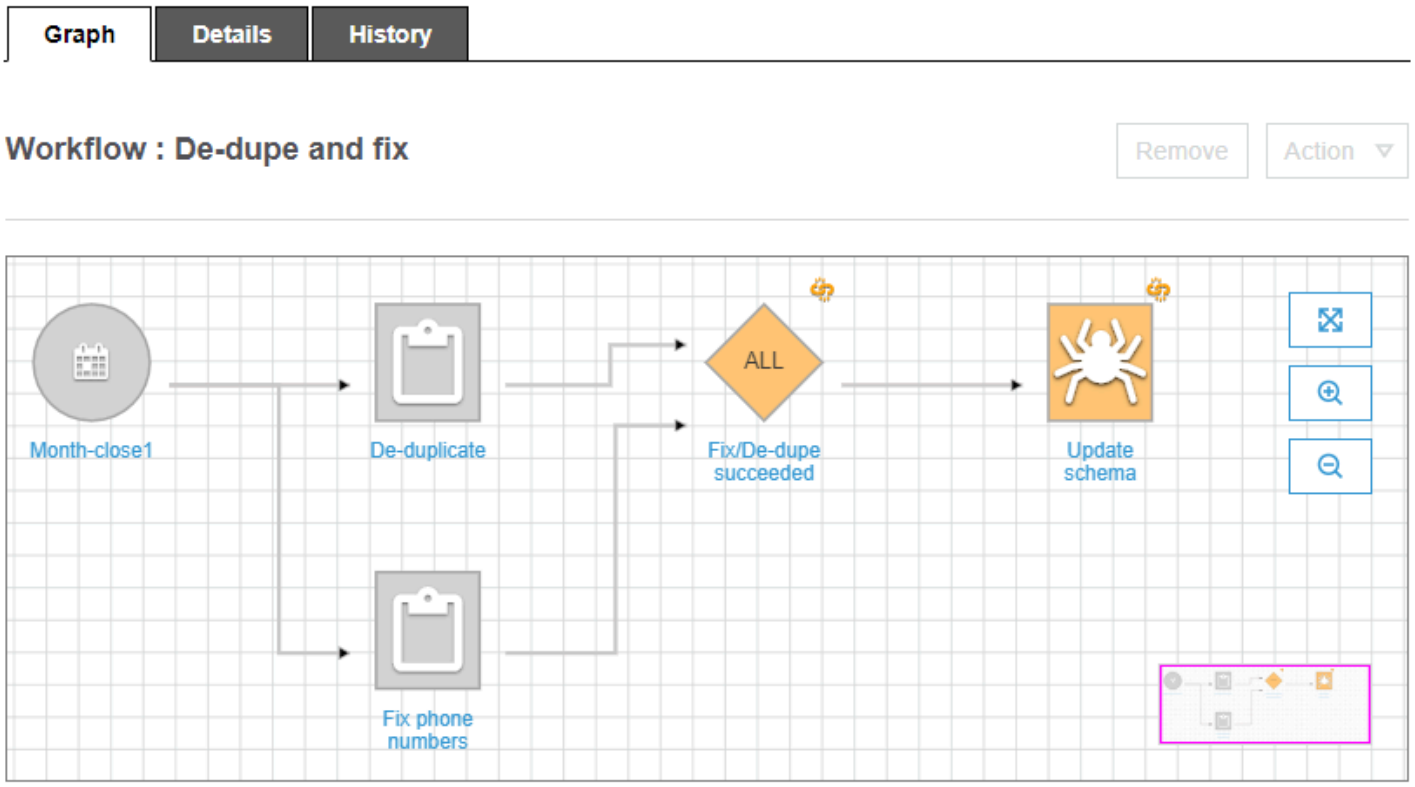
Uma execução de fluxo de trabalho não será iniciada se exceder o limite de simultaneidade definido para o fluxo de trabalho, mesmo que a condição do evento seja atendida. É aconselhável ajustar os limites de simultaneidade do fluxo de trabalho com base no volume de eventos esperado. O AWS Glue não tenta executar novamente o fluxo de trabalho que falha por causa de limites de simultaneidade excedidos. Da mesma forma, é aconselhável ajustar os limites de simultaneidade para trabalhos e crawlers dentro de fluxos de trabalho com base no volume de eventos esperado.

Propriedades de execução do fluxo de trabalho

Para compartilhar e gerenciar estados em uma execução de fluxo de trabalho, defina as propriedades de execução de fluxo de trabalho padrão. Essas propriedades, que são pares de nome/valor, estão disponíveis para todos os trabalhos no fluxo de trabalho. Usando o AWS Glue API, os trabalhos podem recuperar as propriedades de execução do fluxo de trabalho e modificá-las para trabalhos fornecidos posteriormente no fluxo de trabalho.

Grafo do fluxo de trabalho

A imagem a seguir mostra o gráfico de um fluxo de trabalho básico no console do AWS Glue. Seu fluxo de trabalho poderia ter dezenas de componentes.



Esse fluxo de trabalho é iniciado por um acionador de programação, `Month-close1`, que inicia dois trabalhos, `De-duplicate` e `Fix phone numbers`. Após a conclusão bem-sucedida de ambos os trabalhos, um acionador de evento, `Fix/De-dupe succeeded`, inicia um crawler, `Update schema`.

Visualizações estáticas e dinâmicas do fluxo de trabalho

Para cada fluxo de trabalho, existe o conceito de visualização estática e dinâmica. A visualização estática indica o design do fluxo de trabalho. A visualização dinâmica é uma visualização do runtime que inclui as informações mais recentes de execução de cada um dos trabalhos e crawlers. As informações de execução incluem o status de sucesso e os detalhes de erros.

Quando um fluxo de trabalho estiver em execução, o console exibirá a visualização dinâmica graficamente, indicando os trabalhos que foram concluídos e os que ainda não foram executados. Também é possível recuperar uma visualização dinâmica de um fluxo de trabalho em execução usando o AWS Glue API. Para ter mais informações, consulte Consultar fluxos de trabalho usando o AWS Glue API.

Consulte também

- the section called “Criar um fluxo de trabalho com base em um esquema”
- the section called “Criar e desenvolver um fluxo de trabalho manualmente”
- Fluxos de trabalho (para a API de fluxos de trabalho)

Criar e desenvolver um fluxo de trabalho manualmente no AWS Glue

Use o console do AWS Glue para criar e construir manualmente um fluxo de trabalho um nó por vez.

Um fluxo de trabalho contém tarefas, crawlers e gatilhos. Antes de criar manualmente um fluxo de trabalho, crie os trabalhos e crawlers que o fluxo de trabalho deve incluir. É melhor especificar crawlers com execução sob demanda para fluxos de trabalho. É possível criar gatilhos durante o desenvolvimento do fluxo de trabalho ou clonar gatilhos existentes no fluxo de trabalho. Quando você clona um acionador, todos os objetos do catálogo associados ao acionador (os trabalhos ou crawlers que o acionam e os trabalhos ou crawlers que ele inicia) são adicionados ao fluxo de trabalho.

Important

Limite o número total de trabalhos, crawlers e acionadores em um fluxo de trabalho a 100 ou menos. Se você incluir mais de 100, poderá receber erros ao tentar retomar ou interromper as execuções do fluxo de trabalho.

Desenvolva o fluxo de trabalho ao adicionar gatilhos ao gráfico do fluxo de trabalho e ao definir os eventos monitorados e as ações para cada gatilho. Comece com um gatilho de início, que pode ser um gatilho sob demanda ou programado e conclua o gráfico adicionando gatilhos de evento (condicionais).

Etapa 1: Criar o fluxo de trabalho

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em ETL, selecione Workflows (Fluxos de trabalho).
3. Selecione Add workflow (Adicionar fluxo de trabalho) e preencha o formulário Add a new ETL workflow (Adicionar um novo fluxo de trabalho de ETL).

As propriedades de execução padrão opcional que você adicionar são disponibilizadas como argumentos para todas as tarefas no fluxo de trabalho. Para ter mais informações, consulte Obter e configurar as propriedades de execução de fluxo de trabalho no AWS Glue.

4. Selecione Add workflow (Adicionar fluxo de trabalho).

   O novo fluxo de trabalho é exibido na lista da página Workflows (Fluxos de trabalho).

Etapa 2: Adicionar um gatilho de início

1. Na página Workflows (Fluxos de trabalho), selecione o novo fluxo de trabalho. Em seguida, na parte inferior da página, certifique-se de que a guia Graph (Gráfico) esteja selecionada.
2. Selecione Add trigger (Adicionar gatilho) e na caixa de diálogo Add trigger (Adicionar gatilho), execute uma das seguintes ações:

   - Selecione Clone existing (Clonar existente), e escolha um gatilho para ser clonado. Em seguida, escolha Adicionar.

     O gatilho é exibido no gráfico com as tarefas e os crawlers que ele monitora e as tarefas e os crawlers que ele inicia.

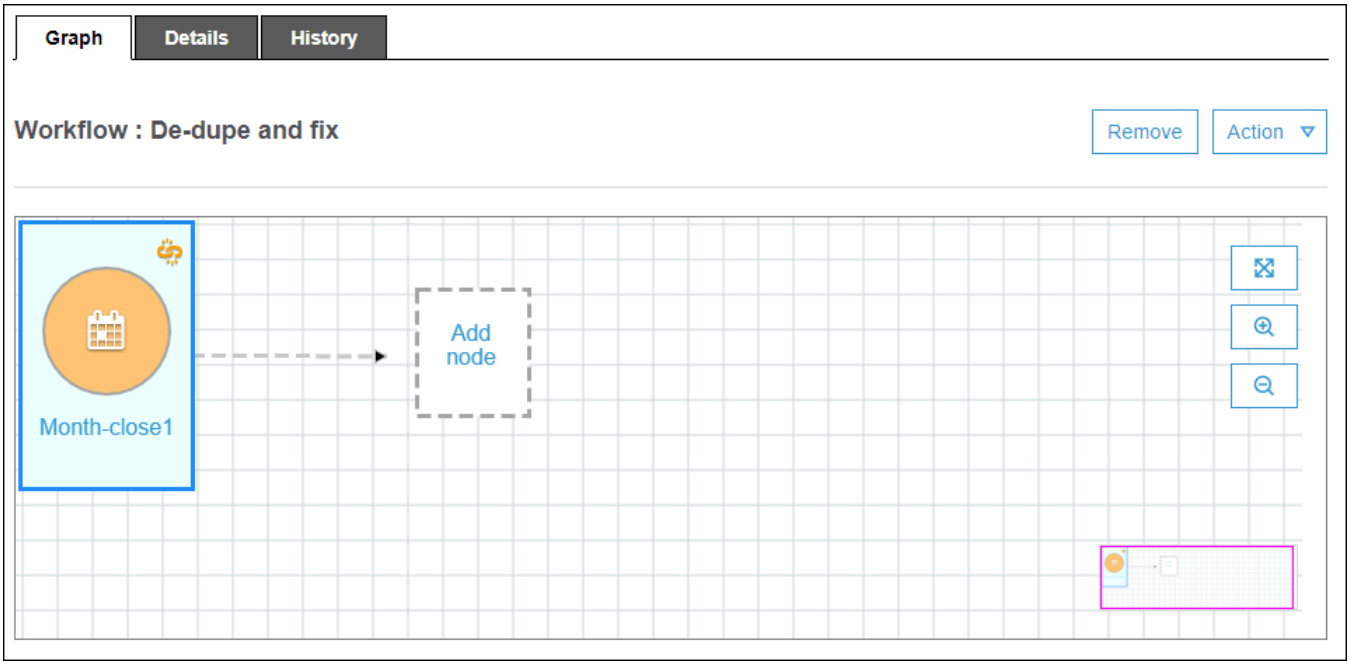
     Se você selecionou o gatilho errado por engano, selecione o gatilho no gráfico e selecione Remove (Remover).
   - Selecione Add new (Adicionar novo) e preencha o formulário Add trigger (Adicionar gatilho).
     1. Para Trigger type (Tipo de acionador), selecione Schedule (Programação), On demand (Sob demanda) ou EventBridge event (Evento do EventBridge).

        Para o tipo de acionador Schedule (Programação), escolha uma das opções de Frequency (Frequência). Escolha Custom (Personalizado) para inserir uma expressão `cron`.

        Para o tipo de acionador EventBridge event (Evento do EventBridge), insira o Number of events (Número de eventos; tamanho do lote) e, opcionalmente, insira o Time delay (Tempo de atraso; janela de lote). Se você omitir o Time delay (Tempo de atraso), o padrão da janela do lote será 15 minutos. Para ter mais informações, consulte Visão geral de fluxos de trabalho no AWS Glue.
     2. Escolha Add.

O gatilho é exibido no gráfico com um nó de espaço reservado (identificado como Add node (Adicionar nó)). No exemplo abaixo, o acionador de início é um acionador de programação chamado `Month-close1`.

Neste momento, o acionador ainda não está salvo.



3. Se você adicionou um novo gatilho, execute estas etapas:

   a. Execute um destes procedimentos:

      - Escolha o nó de espaço reservado (Add node (Adicionar nó)).
      - Certifique-se de que o gatilho de início esteja selecionado e, no menu Action (Ação) acima do gráfico, selecione Add jobs/crawlers to trigger (Adicionar tarefas/crawlers ao gatilho).

   b. Na caixa de diálogo Add job(s) and crawler(s) to trigger (Adicionar tarefas e crawlers ao gatilho), selecione uma ou mais tarefas ou crawlers e selecione Add (Adicionar).

      O gatilho é salvo e as tarefas ou crawlers selecionados são exibidos no gráfico com conectores do gatilho.

      Se você adicionou tarefas ou crawlers errados por engano, pode selecionar o gatilho ou um conector e selecionar Remove (Remover).

Etapa 3: adicionar mais acionadores

Continue a construir o fluxo de trabalho adicionando mais acionadores do tipo Event (Evento). Para aumentar ou diminuir o zoom ou para ampliar a tela do gráfico, use os ícones à direita do gráfico. Para cada gatilho a ser adicionado, conclua as seguintes etapas:

> **Note**
>
> Não há nenhuma ação para salvar o fluxo de trabalho. Depois de adicionar o último acionador e atribuir ações a ele, o fluxo de trabalho será concluído e salvo. Você pode voltar a qualquer momento e adicionar mais nós.

1. Execute um destes procedimentos:

   - Para clonar um gatilho existente, certifique-se de que nenhum nó do gráfico esteja selecionado e, no menu Action (Ação), selecione Add trigger (Adicionar gatilho).
   - Para adicionar um novo gatilho que monitora uma tarefa ou um crawler específico no gráfico, selecione o nó da tarefa ou do crawler e selecione o nó do espaço reservado Add trigger (Adicionar gatilho).

     É possível adicionar mais tarefas ou crawlers a serem monitorados por esse gatilho em uma etapa posterior.
2. Na caixa de diálogo Add trigger (Adicionar gatilho), siga um destes procedimentos:

   - Selecione Add new (Adicionar novo) e preencha o formulário Add trigger (Adicionar gatilho). Em seguida, escolha Adicionar.

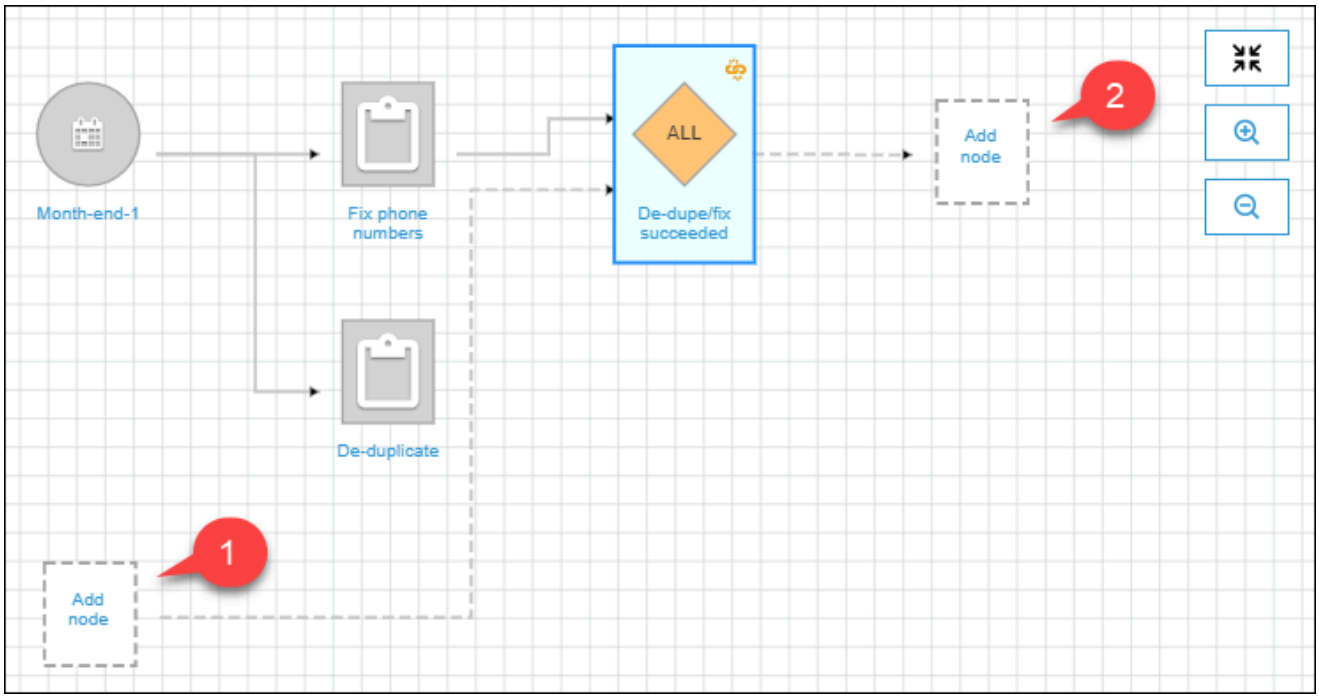
     O gatilho é exibido no gráfico. Você concluirá o gatilho em uma etapa posterior.
   - Selecione Clone existing (Clonar existente), e escolha um gatilho para ser clonado. Em seguida, escolha Adicionar.

     O gatilho é exibido no gráfico com as tarefas e os crawlers que ele monitora e as tarefas e os crawlers que ele inicia.

     Se você escolheu o gatilho errado por engano, selecione-o no gráfico e selecione Remove (Remover).
3. Se você adicionou um novo gatilho, execute estas etapas:

a. Selecione o novo gatilho.

   Como mostra o gráfico a seguir, o acionador `De-dupe/fix succeeded` está selecionado, e os nós de espaço reservado aparecem para (q) evento a ser monitorado e (2) ações.



b. (Opcional caso o gatilho já esteja monitorando um evento e você queira adicionar mais tarefas ou crawlers para serem monitorados.) Escolha o nó de espaço reservado de eventos a serem monitorados e, na caixa de diálogo Add job(s) and crawler(s) to watch (Adicionar tarefas e crawlers a serem monitorados), selecione uma ou mais tarefas ou crawlers. Escolha um evento a ser monitorado (BEM-SUCEDIDO, COM FALHA, etc.) e selecione Add (Adicionar).

c. Certifique-se de que o gatilho esteja selecionado e escolha o nó de espaço reservado de ações.

d. Na caixa de diálogo Add job(s) and crawler(s) to watch (Adicionar tarefas e crawlers a serem monitorados), selecione uma ou mais tarefas e crawlers e selecione Add (Adicionar).

   As tarefas e os crawlers selecionados são exibidos no gráfico com conectores do gatilho.

Para obter mais informações sobre fluxos de trabalho e esquemas, consulte os tópicos a seguir.

- Visão geral de fluxos de trabalho no AWS Glue
- Executar e monitorar um fluxo de trabalho no AWS Glue

- Criar um fluxo de trabalho com base em um esquema no AWS Glue

Iniciar um fluxo de trabalho do AWS Glue com um evento do Amazon EventBridge

O Amazon EventBridge, também conhecido como CloudWatch Events, permite que você automatize seus produtos da AWS e responda automaticamente aos eventos do sistema, como problemas de disponibilidade da aplicação ou alterações de recursos. Os eventos dos serviços da AWS são entregues ao EventBridge quase em tempo real. Você pode escrever regras simples para indicar quais eventos são do seu interesse, e as ações automatizadas a serem tomadas quando um evento corresponder à regra.

Com suporte a EventBridge, o AWS Glue pode servir como produtor e consumidor de eventos em uma arquitetura orientada por eventos. Para fluxos de trabalho, o AWS Glue suporta qualquer tipo de evento do EventBridge como consumidor. O mais provável caso de uso é a chegada de um novo objeto em um bucket do Amazon S3. Se você tiver dados chegando em intervalos irregulares ou indefinidos, você poderá processar esses dados o mais próximo possível de suas chegadas.

> **Note**
>
> O AWS Glue não fornece entrega garantida de mensagens do EventBridge. O AWS Glue não realizará a eliminação de duplicação se o EventBridge fornecer mensagens duplicadas. Você deve gerenciar a idempotência com base no seu caso de uso.
> Certifique-se de configurar as regras do EventBridge corretamente para evitar o envio de eventos indesejados.

Antes de começar

Se você quiser iniciar um fluxo de trabalho com eventos de dados do Amazon S3, certifique-se de que os eventos do bucket do S3 de interesse sejam registrados no AWS CloudTrail e no EventBridge. Para fazer isso, você deve criar uma trilha do CloudTrail. Para obter mais informações, consulte Criar uma trilha para sua conta da AWS.

Para iniciar um fluxo de trabalho com um evento do EventBridge

> **Note**
>
> Nos comandos a seguir, substitua:
>
> - *<workflow-name>* pelo nome a ser atribuído ao fluxo de trabalho.
> - *<trigger-name>* pelo nome a ser atribuído ao acionador.
> - *<bucket-name>* pelo nome do bucket do Amazon S3.
> - *<account-id>* por um ID da conta da AWS válido.
> - *<region>* pelo nome da região (por exemplo, `us-east-1`).
> - *<rule-name>* pelo nome a ser atribuído à regra do EventBridge.

1. Verifique se você tem permissões do AWS Identity and Access Management (IAM) para criar e exibir regras e destinos do EventBridge. Veja a seguir um exemplo de política que você pode anexar. Você pode querer restringir o escopo para impor limites nas operações e 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. Crie uma função do IAM que o produto EventBridge possa assumir ao passar um evento para o AWS Glue.

a. Na página Criar perfil do console do IAM, escolha Serviço da AWS. Em seguida, escolha o produto CloudWatch Events.

b. Conclua o assistente Create role (Criar função). O assistente anexa automaticamente as políticas `CloudWatchEventsBuiltInTargetExecutionAccess` e `CloudWatchEventsInvocationAccess`.

c. Anexe a política em linha a seguir à função. Esta política permite que o produto EventBridge direcione eventos para o AWS Glue.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:notifyEvent"
      ],
      "Resource": [
        "arn:aws:glue:<region>:<account-id>:workflow/<workflow-name>"
      ]
    }
  ]
}
```

3. Insira o comando a seguir para criar o fluxo de trabalho.

   Consulte criar fluxo de trabalho na Referência de comandos da AWS CLI para obter informações sobre mais parâmetros opcionais da linha de comando.

```
aws glue create-workflow --name <workflow-name>
```

4. Insira o seguinte comando para criar um acionador de evento do EventBridge para o fluxo de trabalho. Este será o acionador inicial para o fluxo de trabalho. Substitua *<actions>* pelas ações a serem executadas (os trabalhos e os crawlers a serem iniciados).

   Consulte criar trigger na Referência de comandos da AWS CLI para obter informações sobre como codificar o argumento `actions`.

```
aws glue create-trigger --workflow-name <workflow-name> --type EVENT --
name <trigger-name> --actions <actions>
```

Se desejar que o fluxo de trabalho seja acionado por um lote de eventos em vez de um único evento do EventBridge, insira o seguinte comando.

```
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 o argumento `event-batching-condition`, `BatchSize` é necessário e `BatchWindow` é opcional. Se `BatchWindow` for omitido, o padrão da janela será 900 segundos, que é a duração máxima da janela.

Example

O exemplo a seguir cria um acionador que inicia o fluxo de trabalho `eventtest` após a chegada de três eventos do EventBridge ou cinco minutos após a chegada do primeiro evento, o que ocorrer primeiro.

```
aws glue create-trigger --workflow-name eventtest --type EVENT --name objectArrival
 --event-batching-condition BatchSize=3,BatchWindow=300 --actions JobName=test1
```

5. Crie uma regra no Amazon EventBridge.

   a. Crie o objeto JSON para os detalhes da regra no editor de texto de sua preferência.

      O exemplo a seguir especifica o Amazon S3 como a origem do evento, `PutObject` como o nome do evento e o nome do bucket como um parâmetro de solicitação. Essa regra inicia um fluxo de trabalho quando um novo objeto chega ao bucket.

```
{
  "source": [
    "aws.s3"
  ],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3.amazonaws.com"
    ],
    "eventName": [
```

```
      "PutObject"
    ],
    "requestParameters": {
      "bucketName": [
        "<bucket-name>"
      ]
    }
  }
}
```

Para iniciar o fluxo de trabalho quando um novo objeto chega a uma pasta dentro do bucket, você pode substituir o seguinte código por `requestParameters`.

```
    "requestParameters": {
      "bucketName": [
        "<bucket-name>"
      ]
      "key" : [{ "prefix" : "<folder1>/<folder2>/*"}}]
  }
```

b. Use sua ferramenta preferida para converter o objeto JSON de regra em uma string 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. Execute o seguinte comando para criar um modelo de parâmetro JSON que você pode editar para especificar parâmetros de entrada para um comando `put-rule`. Salve a saída em um arquivo. Para este exemplo, o arquivo é chamado `ruleCommand`.

```
aws events put-rule --name <rule-name> --generate-cli-skeleton >ruleCommand
```

Para obter mais informações sobre o parâmetro `--generate-cli-skeleton`, consulte Gerar um esqueleto da AWS CLI e parâmetros de entrada usando um arquivo de entrada JSON ou YAML no Manual do usuário da interface da AWS Command Line.

O arquivo de saída deve ser como o a seguir.

```
{
    "Name": "",
    "ScheduleExpression": "",
    "EventPattern": "",
    "State": "ENABLED",
    "Description": "",
    "RoleArn": "",
    "Tags": [
        {
            "Key": "",
            "Value": ""
        }
    ],
    "EventBusName": ""
}
```

d. Edite o arquivo para opcionalmente remover parâmetros e especificar, no mínimo, os parâmetros `Name`, `EventPattern` e `State`. Para o parâmetro `EventPattern`, forneça a string de escape para os detalhes da regra que você criou em uma etapa 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**
>
> É melhor deixar a regra desabilitada até terminar de criar o fluxo de trabalho.

e. Insira o seguinte comando `put-rule`, que lê os parâmetros de entrada do arquivo `ruleCommand`.

```
aws events put-rule --name <rule-name> --cli-input-json file://ruleCommand
```

A seguinte saída indica sucesso.

```
{
    "RuleArn": "<rule-arn>"
}
```

6. Insira o comando a seguir para anexar a regra a um destino. O destino é o fluxo de trabalho no AWS Glue. Substitua *<role-name>* pela função criada por você no início deste procedimento.

```
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>
```

A seguinte saída indica sucesso.

```
{
    "FailedEntryCount": 0,
    "FailedEntries": []
}
```

7. Confirme a conexão bem-sucedida da regra e do destino inserindo o seguinte comando.

```
aws events list-rule-names-by-target --target-arn arn:aws:glue:<region>:<account-
id>:workflow/<workflow-name>
```

A seguinte saída indica sucesso, em que *<rule-name>* é o nome da regra que você criou.

```
{
    "RuleNames": [
        "<rule-name>"
    ]
}
```

8. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.

9. Selecione o fluxo de trabalho e verifique se o acionador inicial e suas ações, os trabalhos ou crawlers que ele inicia, aparecem no gráfico do fluxo de trabalho. Em seguida, vá para o procedimento em Etapa 3: adicionar mais acionadores. Ou adicione mais componentes ao fluxo de trabalho usando a API do AWS Glue ou AWS Command Line Interface.

10. Quando o fluxo de trabalho estiver completamente especificado, habilite a regra.

```
aws events enable-rule --name <rule-name>
```

O fluxo de trabalho agora está pronto para ser iniciado por um evento do EventBridge ou lote de eventos.

Consulte também

- Manual do usuário do Amazon EventBridge
- Visão geral de fluxos de trabalho no AWS Glue
- Criar e desenvolver um fluxo de trabalho manualmente no AWS Glue

Visualizar os eventos do EventBridge que iniciaram um fluxo de trabalho

Você pode visualizar o ID do evento do Amazon EventBridge que iniciou seu fluxo de trabalho. Se o fluxo de trabalho foi iniciado por um lote de eventos, você pode exibir os IDs de evento de todos os eventos no lote.

Para fluxos de trabalho com um tamanho de lote maior que um, você também pode ver qual condição de lote iniciou o fluxo de trabalho: a chegada do número de eventos no tamanho do lote ou a validade da janela do lote.

Para exibir os eventos do EventBridge que iniciaram um fluxo de trabalho (console)

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Workflows (Fluxos de trabalho).
3. Selecione o fluxo de trabalho. Em seguida, na parte inferior, escolha a guia History (Histórico).
4. Selecione uma execução de fluxo de trabalho e escolha View run details (Visualizar detalhes da execução).
5. Na página de detalhes da execução, localize o campo Run properties (Propriedades de execução) e procure a chave aws:eventIds.

   O valor dessa chave é uma lista de IDs de eventos do EventBridge.

Para exibir os eventos do EventBridge que iniciaram um fluxo de trabalho (API da AWS)

- Inclua o seguinte código em seu script 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 uma lista de strings, em que cada string é um ID de evento.

> Consulte também
>
> - Manual do usuário do Amazon EventBridge
> - the section called “Visão geral de fluxos de trabalho”

Executar e monitorar um fluxo de trabalho no AWS Glue

Se o gatilho de início de um fluxo de trabalho for um gatilho sob demanda, será possível iniciar o fluxo de trabalho no console do AWS Glue. Conclua as etapas a seguir para executar e monitorar um fluxo de trabalho. Se o fluxo de trabalho falhar, é possível visualizar o gráfico de execução para determinar o nó que falhou. Para ajudar a solucionar problemas, se o fluxo de trabalho tiver sido criado de um blueprint, você poderá exibir a execução do blueprint para ver os valores dos parâmetros do blueprint usados para criar o fluxo de trabalho. Para ter mais informações, consulte the section called “Visualizar execuções de esquemas”.

Você executar e monitorar um fluxo de trabalho usando o console do AWS Glue, API ou o AWS Command Line Interface (AWS CLI).

Para executar e monitorar um fluxo de trabalho (console)

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em ETL, selecione Workflows (Fluxos de trabalho).
3. Selecione um fluxo de trabalho. No menu Actions (Ações), selecione Run (Executar).
4. Confira a coluna Last run status (Status da última execução) na lista de fluxos de trabalho. Escolha o botão de atualização para exibir o status do fluxo de trabalho em andamento.

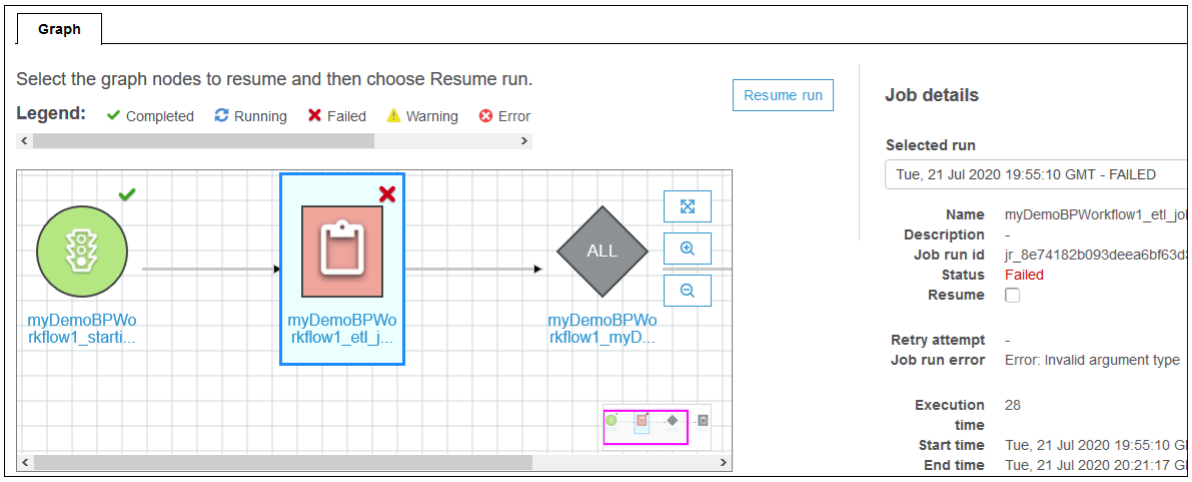
5. Enquanto o fluxo de trabalho estiver em execução, ou após a conclusão (ou falha), visualize os detalhes da execução concluindo as etapas a seguir.

   a. Certifique-se de que o fluxo de trabalho esteja selecionado e escolha a guia History (Histórico).

   b. Escolha a execução do fluxo de trabalho atual ou mais recente e escolha View run details (Visualizar detalhes da execução).

      O gráfico de runtime do fluxo de trabalho mostra o status de execução atual.

   c. Escolha qualquer nó no gráfico para exibir os detalhes e status dele.



Para executar e monitorar um fluxo de trabalho (AWS CLI)

1. Insira o comando a seguir. Substitua *<workflow-name>* pelo fluxo de trabalho a ser executado.

```
aws glue start-workflow-run --name <workflow-name>
```

   Se o fluxo de trabalho for iniciado com êxito, o comando retornará o ID de execução.

2. Visualize o status de execução do fluxo de trabalho usando o comando `get-workflow-run`. Forneça o nome do fluxo de trabalho e o ID de execução.

```
aws glue get-workflow-run --name myWorkflow --run-id
 wr_d2af14217e8eae775ba7b1fc6fc7a42c795aed3cbcd8763f9415452e2dbc8705
```

   Este é um exemplo de saída de 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
        }
    }
}
```

Consulte também:

- the section called “Visão geral de fluxos de trabalho”
- the section called “Visão geral dos esquemas”

Interromper uma execução de fluxo de trabalho

É possível usar o console do AWS Glue, a AWS Command Line Interface (AWS CLI) ou a API do AWS Glue para interromper uma execução de fluxo de trabalho. Quando você interrompe uma execução de fluxo de trabalho, todos os crawlers e os trabalhos em execução são imediatamente encerrados, e os que ainda não foram iniciados nunca serão. Pode levar até um minuto para que todos os crawlers e os trabalhos em execução sejam interrompidos. O status de execução do fluxo de trabalho passa de Em execução para Interrompendo e, quando a execução do fluxo de trabalho é completamente interrompida, o status passa para Interrompido.

Depois que a execução do fluxo de trabalho é interrompida, é possível visualizar o gráfico de execução para saber quais trabalhos e crawlers foram concluídos e quais nunca começaram. Assim, você pode determinar se é necessário executar alguma etapa para garantir a integridade dos dados. Interromper uma execução de fluxo de trabalho faz com que nenhuma operação de reversão automática seja executada.

Como interromper uma execução de fluxo de trabalho (console)

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, em ETL, selecione Workflows (Fluxos de trabalho).
3. Escolha um fluxo de trabalho em execução e selecione a guia Histórico.
4. Escolha a execução do fluxo de trabalho e selecione Interromper execução.

   O status da execução muda para Interrompendo.
5. (Opcional) Escolha a execução do fluxo de trabalho, selecione Visualizar detalhes da execução e revise o gráfico de execução.

Como interromper uma execução de fluxo de trabalho (AWS CLI)

- Insira o comando a seguir. Substitua *<workflow-name>* pelo nome do fluxo de trabalho e *<run-id>* pelo ID da execução do fluxo de trabalho a ser interrompida.

  ```
  aws glue stop-workflow-run --name <workflow-name> --run-id <run-id>
  ```

  Veja a seguir um exemplo do comando stop-workflow-run.

  ```
  aws glue stop-workflow-run --name my-workflow --run-id
   wr_137b88917411d128081069901e4a80595d97f719282094b7f271d09576770354
  ```

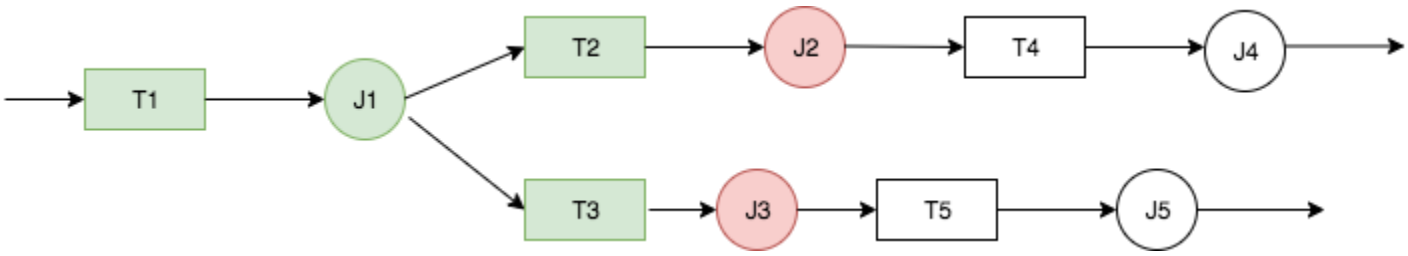
Reparar e retomar uma execução de fluxo de trabalho

Se um ou mais nós (trabalhos ou crawlers) em um fluxo de trabalho não forem concluídos com êxito, isso significa que o fluxo de trabalho foi executado apenas parcialmente. Depois de localizar as causas raiz e fazer correções, você pode selecionar um ou mais nós de onde retomar a execução do fluxo de trabalho e, em seguida, retomá-la. Os nós selecionados e todos os nós que estão a jusante desses nós são então executados.

Tópicos

Retomar uma execução de fluxo de trabalho: como funciona

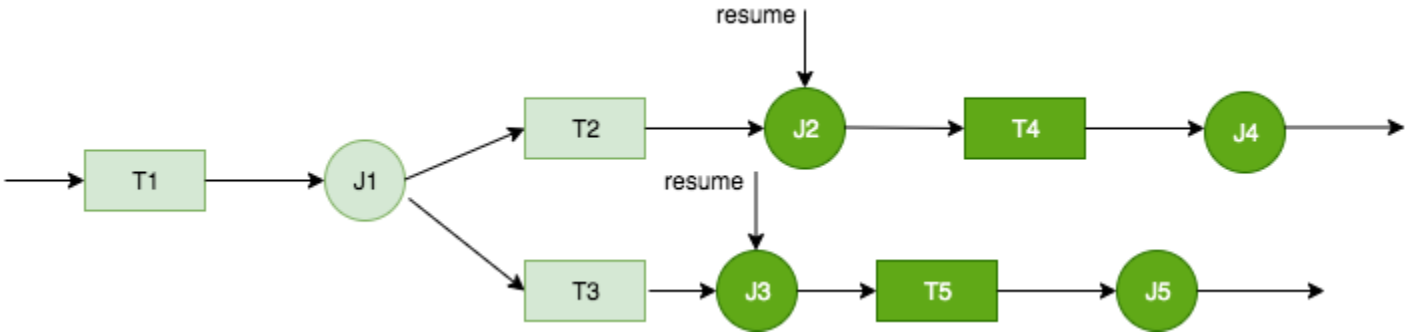
Considere o fluxo de trabalho W1 no diagrama a seguir.



A execução do fluxo de trabalho prossegue da seguinte forma:

1. O acionador T1 inicia o trabalho J1.
2. A conclusão bem-sucedida de J1 aciona T2 e T3, que executam os trabalhos J2 e J3, respectivamente.
3. Os trabalhos J2 e J3 falham.
4. Os acionadores T4 e T5 dependem da conclusão bem-sucedida de J2 e J3, então eles não disparam, e os trabalhos J4 e J5 não são executados. O fluxo de trabalho W1 é executado apenas parcialmente.

Agora, suponha que os problemas que causaram a falha de J2 e J3 sejam corrigidos. J2 e J3 são selecionados como os pontos de partida para retomar a execução do fluxo de trabalho.



A execução do fluxo de trabalho é retomada da seguinte forma:

1. Os trabalhos J2 e J3 são executados com êxito.
2. Os acionadores T4 e T5 disparam.
3. Os trabalhos J4 e J5 são executados com êxito.

A execução do fluxo de trabalho retomado é rastreada como uma execução de fluxo de trabalho separada com um novo ID de execução. Ao exibir o histórico do fluxo de trabalho, você pode exibir o ID de execução anterior para qualquer execução do fluxo de trabalho. No exemplo na captura de tela a seguir, o fluxo de trabalho executado com o ID de execução `wr_c7a22...` (a segunda linha) tinha um nó que não foi concluído. O usuário corrigiu o problema e retomou a execução do fluxo de trabalho, o que resultou no ID de execução `wr_a07e55...` (a primeira linha).

> **Note**
>
> Para o resto desta discussão, o termo “execução do fluxo de trabalho retomado” refere-se à execução do fluxo de trabalho que foi criada quando a execução do fluxo de trabalho anterior foi retomada. A “execução do fluxo de trabalho original” refere-se à execução do fluxo de trabalho que foi executada apenas parcialmente e que precisava ser retomada.

Grafo de execução de fluxo de trabalho retomada

Em uma execução de fluxo de trabalho retomada, embora apenas um subconjunto de nós seja executado, o gráfico de execução é um gráfico completo. Ou seja, os nós que não foram executados no fluxo de trabalho retomado são copiados do gráfico de execução da execução do fluxo de trabalho original. Os nós de trabalho e crawler copiados executados na execução do fluxo de trabalho original incluem os detalhes da execução.

Considere novamente o fluxo de trabalho W1 no diagrama anterior. Quando a execução do fluxo de trabalho é retomada começando com J2 e J3, o gráfico para a execução do fluxo de trabalho retomado mostra todos os trabalhos, J1 a J5, e todos os acionadores, T1 a T5. Os detalhes da execução para J1 são copiados da execução do fluxo de trabalho original.

Snapshots da execução do fluxo de trabalho

Quando uma execução de fluxo de trabalho é iniciada, o AWS Glue tira um snapshot do gráfico de design do fluxo de trabalho naquele ponto no tempo. Esse snapshot é usado durante a execução do fluxo de trabalho. Se você fizer alterações em quaisquer acionadores após o início da execução, essas alterações não afetarão a execução do fluxo de trabalho atual. Os snapshots garantem que as execuções do fluxo de trabalho prossigam de maneira consistente.

Os snapshots tornam apenas os acionadores imutáveis. As alterações feitas em trabalhos e crawlers a jusante durante a execução do fluxo de trabalho entram em vigor para a execução atual.

Retomar uma execução de fluxo de trabalho

Siga estas etapas para retomar uma execução de fluxo de trabalho. Você pode retomar uma execução de fluxo de trabalho usando o console do AWS Glue, a API ou o AWS Command Line Interface (AWS CLI).

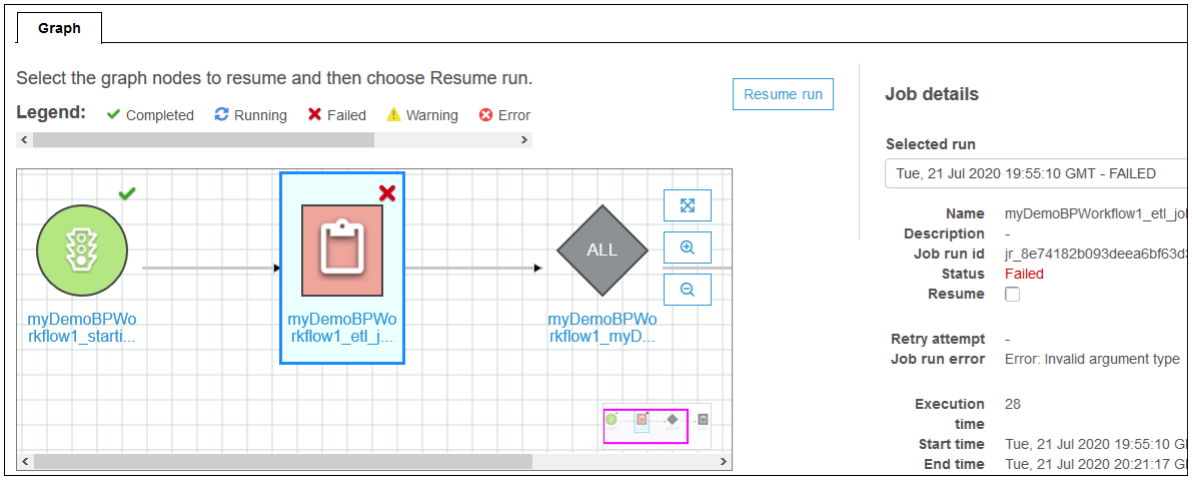
Como retomar a execução de um fluxo de trabalho (console)

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.

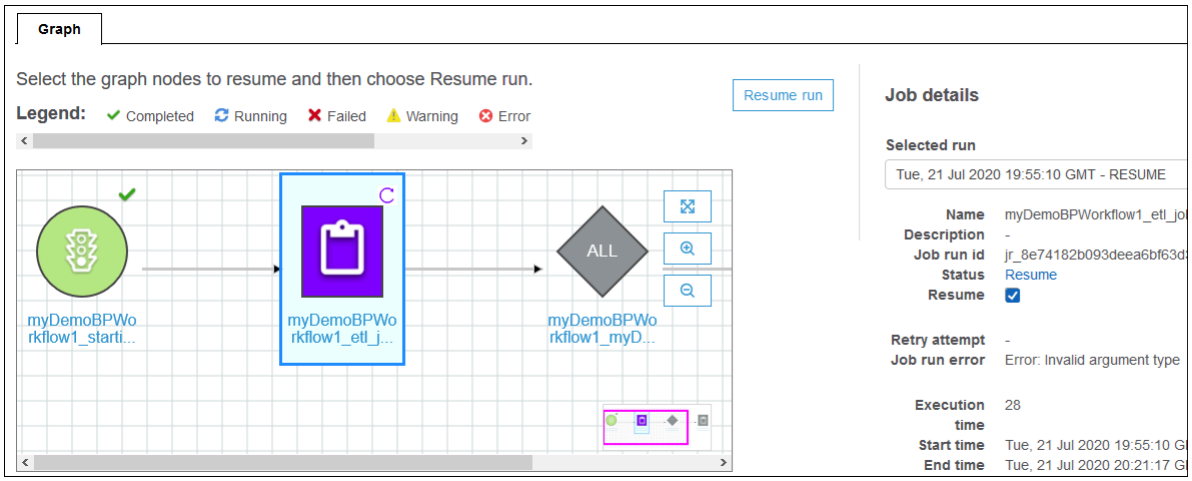
   Faça login como um usuário que tem permissões para exibir fluxos de trabalho e retomar execuções de fluxo de trabalho.

   > **Note**
   >
   > Para retomar execuções de fluxo de trabalho, você precisa da permissão `glue:ResumeWorkflowRun` do AWS Identity and Access Management (IAM).

2. No painel de navegação, escolha Workflows (Fluxos de trabalho).
3. Selecione um fluxo de trabalho e escolha a guia History (Histórico).
4. Selecione a execução do fluxo de trabalho que foi executada apenas parcialmente e escolha View run details (Visualizar detalhes da execução).
5. No gráfico de execução, selecione o primeiro (ou único) nó que deseja reiniciar e do qual deseja retomar a execução do fluxo de trabalho.
6. No painel de detalhes à direita do gráfico, selecione a caixa de seleção Resume (Retomar).



O nó muda de cor e mostra um pequeno ícone de retomada no canto superior direito.



7. Conclua as duas etapas anteriores para que todos os nós adicionais reiniciem.
8. Selecione Resume run (Retomar execução).

Para retomar a execução de um fluxo de trabalho (AWS CLI)

1. Verifique se você tem a permissão `glue:ResumeWorkflowRun` do IAM.
2. Recupere os IDs de nós dos nós que você deseja reiniciar.

   a. Execute o comando `get-workflow-run` para a execução do fluxo de trabalho original. Forneça o nome do fluxo de trabalho e o ID de execução, e adicione a opção `--include-graph`, conforme mostrado no exemplo a seguir. Obtenha o ID de execução na guia History (Histórico) no console, ou executando o comando `get-workflow`.

```
aws glue get-workflow-run --name cloudtrailtest1 --run-id
 wr_a07e55f2087afdd415a404403f644a4265278f68b13ba3da08c71924ebe3c3a8 --include-
graph
```

O comando retorna os nós e as bordas do gráfico como um objeto JSON grande.

b. Localize os nós de interesse pelas propriedades `Type` e `Name` dos objetos de nó.

A seguir, está um exemplo de objeto de nó da saída.

```
{
    "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. Obtenha o ID do nó da propriedade `UniqueId` do objeto de nó.

3. Execute o comando `resume-workflow-run`. Forneça o nome do fluxo de trabalho, o ID da execução e a lista de IDs de nós separados por espaços, conforme mostrado no exemplo a seguir.

```
aws glue resume-workflow-run --name cloudtrailtest1 --run-id
 wr_a07e55f2087afdd415a404403f644a4265278f68b13ba3da08c71924ebe3c3a8 --node-
ids wnode_ca1f63e918fb855e063aed2f42ec5762ccf71b80082ae2eb5daeb8052442f2f3
 wnode_d1b2563c503078b153142ee76ce545fe5ceef66e053628a786ddd74a05da86fd
```

O comando gera o ID de execução do fluxo de trabalho retomado (novo) e uma lista de nós que serão iniciados.

```
{
    "RunId": "wr_2ada0d3209a262fc1156e4291134b3bd643491bcfb0ceead30bd3e4efac24de9",
    "NodeIds": [
        "wnode_ca1f63e918fb855e063aed2f42ec5762ccf71b80082ae2eb5daeb8052442f2f3"
    ]
}
```

Note que, apesar do comando de exemplo `resume-workflow-run` ter listado dois nós para reiniciar, a saída de exemplo indicou que apenas um nó seria reiniciado. Isso ocorre porque um nó estava a jusante do outro, e o nó a jusante seria reiniciado de qualquer maneira pelo fluxo normal do fluxo de trabalho.

Observações e limitações para retomar execuções de fluxo de trabalho

Tenha em mente as seguintes observações e limitações ao retomar as execuções do fluxo de trabalho.

- Você pode retomar uma execução de fluxo de trabalho somente se ele estiver no estado `COMPLETED`.

  > **Note**
  >
  > Mesmo que um ou mais nós em uma execução de fluxo de trabalho não sejam concluídos, o estado de execução do fluxo de trabalho é mostrado como `COMPLETED`. Certifique-se de verificar o gráfico de execução para descobrir todos os nós que não foram concluídos com êxito.

- Você pode retomar uma execução de fluxo de trabalho de qualquer nó de trabalho ou crawler que a execução do fluxo de trabalho original tenha tentado executar. Não é possível retomar uma execução de fluxo de trabalho de um nó de acionador.

- Reiniciar um nó não redefine seu estado. Todos os dados que foram parcialmente processados não são revertidos.
- Você pode retomar a mesma execução de fluxo de trabalho várias vezes. Se um fluxo de trabalho retomado for executado apenas parcialmente, você poderá resolver o problema e retomar a execução retomada.
- Se você selecionar dois nós a serem reiniciados e eles dependerem um do outro, o nó a montante será executado antes do nó a jusante. Na verdade, selecionar o nó a montante é redundante, pois ele será executado de acordo com o fluxo normal do fluxo de trabalho.

Obter e configurar as propriedades de execução de fluxo de trabalho no AWS Glue

Use as propriedades da execução de fluxo de trabalho para compartilhar e gerenciar o estado entre os trabalhos no seu fluxo de trabalho do AWS Glue. Você pode definir as propriedades padrão da execução quando criar o fluxo de trabalho. Depois, enquanto os trabalhos são executados, eles podem recuperar os valores de propriedade da execução e, opcionalmente, modificá-los para a entrada de trabalhos que estarão no fluxo de trabalho mais tarde. Quando um trabalho modifica uma propriedade da execução, o novo valor existe somente para a execução do fluxo de trabalho. As propriedades de execução padrão não são afetadas.

Se sua tarefa do AWS Glue não fizer parte de um fluxo de trabalho, essas propriedades não serão definidas.

O seguinte exemplo de código Python de um trabalho de extração, transformação e carregamento (ETL) demonstra como obter as propriedades da execução do fluxo de trabalho.

```
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']
```

O código a seguir continua definindo a propriedade da execução `target_format` como `'csv'`.

```
workflow_params['target_format'] = 'csv'
glue_client.put_workflow_run_properties(Name=workflow_name, RunId=workflow_run_id,
 RunProperties=workflow_params)
```

Para mais informações, consulte:

- Ação GetWorkflowRunProperties (Python: get_workflow_run_properties)
- Ação PutWorkflowRunProperties (Python: put_workflow_run_properties)

Consultar fluxos de trabalho usando o AWS Glue API

O AWS Glue fornece uma API avançada para gerenciar fluxos de trabalho. É possível recuperar uma visualização estática do fluxo de trabalho ou uma visualização dinâmica de um fluxo de trabalho em execução usando o AWS Glue API. Para ter mais informações, consulte Fluxos de trabalho.

Tópicos

- Consultar visualizações estáticas
- Consultar visualizações dinâmicas

Consultar visualizações estáticas

Use a operação de API `GetWorkflow` para obter uma visualização estática que indica o design de um fluxo de trabalho. Essa operação retorna um gráfico direcionado que consiste em nós e bordas, onde um nó representa um gatilho, uma tarefa ou um crawler. As bordas definem as relações entre os nós. Eles são representados por conectores (setas) no gráfico no console do AWS Glue.

Também é possível usar essa operação com bibliotecas populares de processamento de gráficos, como NetworkX, igraph, JGraphT e a Java Universal Network/Graph (JUNG) Framework. Como todas essas bibliotecas representam gráficos de forma semelhante, são necessárias transformações mínimas.

A visualização estática retornada por essa API é a visualização mais atualizada de acordo com a definição mais recente de gatilhos associados ao fluxo de trabalho.

Definição do grafo

Um gráfico do fluxo de trabalho G é um par ordenado (N, E), onde N é um conjunto de nós e E, um conjunto de bordas. O nó é um vértice no gráfico identificado por um número exclusivo. Um nó pode ser do tipo acionador, de trabalho ou crawler. Por exemplo: `{name:T1, type:Trigger, uniqueId:1}, {name:J1, type:Job, uniqueId:2}`.

A borda é uma tupla de pares do formulário (`src, dest`), onde `src` e `dest` são nós e há uma borda direcionada de `src` para `dest`.

Exemplo de como consultar uma visualização estática

Considere um gatilho condicional T, que aciona a tarefa J2 após a conclusão da tarefa J1.

```
J1 ---> T ---> J2
```

Nós: J1, T, J2

Bordas: (J1, T), (T, J2)

Consultar visualizações dinâmicas

Use a operação de API `GetWorkflowRun` para obter uma visualização dinâmica de um fluxo de trabalho em execução. Essa operação retorna a mesma visualização estática do gráfico com os metadados relacionados à execução do fluxo de trabalho.

Para a execução, nós que representam trabalhos na chamada `GetWorkflowRun` têm uma lista de execuções de trabalhos iniciadas como parte da última execução do fluxo de trabalho. É possível usar essa lista para exibir o status de execução de cada tarefa no próprio gráfico. Para dependências downstream que ainda não foram executadas, esse campo é definido como `null`. As informações no gráfico deixam você ciente do estado atual de qualquer fluxo de trabalho em determinado momento.

A visualização dinâmica retornada por essa API é baseada na visualização estática que estava presente quando a execução do fluxo de trabalho foi iniciada.

Exemplo de nós de runtime: `{name:T1, type: Trigger, uniqueId:1}`, `{name:J1, type:Job, uniqueId:2, jobDetails:{jobRuns}}`, `{name:C1, type:Crawler, uniqueId:3, crawlerDetails:{crawls}}`

Exemplo 1: visualização dinâmica

O exemplo a seguir ilustra um fluxo de trabalho simples de dois gatilhos.

- Nós: t1, j1, t2, j2
- Bordas: (t1, j1), (j1, t2), (t2, j2)

A resposta `GetWorkflow` contém o seguinte.

```
{
    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
        }
}
```

A resposta `GetWorkflowRun` contém o seguinte.

```
{
    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
        }
}
```

Exemplo 2: vários trabalhos com um acionador condicional

O exemplo a seguir mostra um fluxo de trabalho com várias tarefas e um gatilho 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)
```

Restrições de esquema e fluxo de trabalho no AWS Glue

Veja a seguir as restrições de blueprint e fluxos de trabalho.

Restrições de esquema

Tenha em mente as seguintes restrições de blueprint:

- O blueprint deve ser registrado na mesma região da AWS em que reside o bucket do Amazon S3.
- Para compartilhar blueprints entre contas da AWS, você deve conceder as permissões de leitura no arquivo ZIP do blueprint no Amazon S3. Os clientes que tiverem permissão de leitura em um arquivo ZIP de blueprint podem registrar o blueprint em suas contas da AWS e usá-lo.
- O conjunto de parâmetros do blueprint é armazenado como um único objeto JSON. O comprimento máximo deste objeto é de 128 KB.
- O tamanho máximo do arquivo ZIP do blueprint descompactado é de 5 MB. O tamanho máximo compactado é de 1 MB.
- Limite o número total de trabalhos, crawlers e acionadores em um fluxo de trabalho a 100 ou menos. Se você incluir mais de 100, poderá receber erros ao tentar retomar ou interromper as execuções do fluxo de trabalho.

Restrições do fluxo de trabalho

Tenha em mente as seguintes restrições do fluxo de trabalho: Alguns desses comentários são mais direcionados a um usuário que esteja criando fluxos de trabalho manualmente.

- O tamanho máximo do lote para um acionador de evento do Amazon EventBridge é 100. O valor máximo da janela é 900 segundos (15 minutos).
- Um gatilho pode ser associado a somente um fluxo de trabalho.
- Somente um gatilho de início (sob demanda ou programação) é permitido.
- Se um trabalho ou crawler em um fluxo de trabalho for iniciado por um gatilho que está fora do fluxo de trabalho, nenhum gatilho interno do fluxo de trabalho que depender da conclusão do trabalho ou do crawler (com êxito ou não) será acionado.
- Da mesma forma, se um trabalho ou crawler em um fluxo de trabalho tiver acionadores que dependam da conclusão do trabalho ou do crawler (com êxito ou não), tanto dentro quanto fora do fluxo de trabalho, e se o trabalho ou o crawler for iniciado dentro de um fluxo de trabalho, somente os acionadores internos desse fluxo de trabalho serão acionados após a conclusão do trabalho ou do crawler.

Solucionar erros de esquema no AWS Glue

Se você encontrar erros ao usar blueprints do AWS Glue, utilize as seguintes soluções para ajudar a encontrar a fonte dos problemas e corrigi-los.

Tópicos

Erro: módulo PySpark ausente

O AWS Glue retorna o erro “Unknown error executing layout generator function ModuleNotFoundError: No module named ‘pyspark’” (Erro desconhecido ao executar a função do gerador de layout ModuleNotFounderror: nenhum módulo chamado “pyspark”).

Ao descompactar o arquivo de blueprint, ele pode ser como um dos seguintes:

```
$ 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
```

No primeiro caso, todos os arquivos relacionados ao blueprint foram colocados em uma pasta chamada compactação e foi então convertido em um arquivo zip chamado compaction.zip.

No segundo caso, todos os arquivos necessários para o blueprint não foram incluídos em uma pasta e foram adicionados como arquivos raiz no arquivo zip compaction.zip.

Criar um arquivo em qualquer um dos formatos acima é permitido. No entanto, certifique-se de que `blueprint.cfg` tenha o caminho correto para o nome da função no script que gera o layout.

Exemplos

No caso 1: `blueprint.cfg` deve ter `layoutGenerator` como o seguinte:

```
layoutGenerator": "compaction.layout.generate_layout"
```

No caso 2: `blueprint.cfg` deve ter `layoutGenerator` como o seguinte:

```
layoutGenerator": "layout.generate_layout"
```

Se esse caminho não estiver incluído corretamente, você poderá ver um erro conforme indicado. Por exemplo, se tiver a estrutura de pastas conforme mencionado no caso 2 e tiver `layoutGenerator` indicado como no caso 1, você poderá ver o erro acima.

Erro: arquivo de configuração do blueprint ausente

O AWS Glue retorna o erro “Unknown error executing layout generator function FileNotFoundError: [Errno 2] No such file or directory: ‘/tmp/compaction/blueprint.cfg’” (Erro desconhecido executando a função do gerador de layout FileNotFoundError: [Errno 2] Nenhum arquivo ou diretório: “/tmp/compaction/blueprint.cfg”).

O blueprint.cfg deve ser colocado no nível raiz do arquivo ZIP ou dentro de uma pasta que tenha o mesmo nome que o arquivo ZIP.

Quando extraímos o arquivo ZIP do blueprint, blueprint.cfg deve ser encontrado em um dos seguintes caminhos. Se não for encontrado em um dos seguintes caminhos, você poderá ver o erro acima.

```
$ unzip compaction.zip
Archive:  compaction.zip
```

```
   creating: compaction/
  inflating: compaction/blueprint.cfg

$ unzip compaction.zip
Archive:  compaction.zip
  inflating: blueprint.cfg
```

Erro: arquivo importado ausente

O AWS Glue retorna o erro “Unknown error executing layout generator function FileNotFoundError: [Errno 2] No such file or directory: ‘demo-project/foo.py’” (Erro desconhecido executando a função do gerador de layout FileNotFoundError: [Errno 2] Nenhum arquivo ou diretório: “demo-project/foo.py”).

Se o script de geração de layout tiver funcionalidade para ler outros arquivos, certifique-se de fornecer um caminho completo para que o arquivo seja importado. Por exemplo, o script Conversion.py pode ser referenciado em Layout.py. Para obter mais informações, consulte Projeto de esquema de exemplo.

Error: not authorized to perform iamPassRole on resource (Erro: não autorizado a executar iamPassRole no recurso)

O AWS Glue retorna o erro “User: arn:aws:sts:: 123456789012:Assumed-Role/AWSGlueservicerole/Gluesession is not authorized to perform: iam:PassRole on resource: arn:aws:iam:: 123456789012:Role/AWSGlueservicerole” (Usuário: arn:aws:sts:: 123456789012:Assumed-Role/AWSGlueservicerole/Gluesession não está autorizado a realizar: iam:PassRole no recurso: arn:aws:iam:: 123456789012:Role/AWSGlueservicerole)

Se os trabalhos e os crawlers no fluxo de trabalho assumirem a mesma função que aquela passada para criar o fluxo de trabalho do blueprint, a função do blueprint precisará incluir a permissão `iam:PassRole` em si mesma.

Se os trabalhos e os crawlers no fluxo de trabalho assumirem uma função diferente da função passada para criar as entidades do fluxo de trabalho do blueprint, a função do blueprint precisará incluir a propriedade `iam:PassRole` nessa outra função em vez de na função do blueprint.

Para obter mais informações, consulte Permissões para perfis de esquema.

Erro: programação cron. inválida

O AWS Glue retorna o erro “The schedule cron(0 0 * * * *) is invalid” (A programação cron(0 0 * * * *) é inválida).

Forneça uma expressão cron. válida. Para obter mais informações, consulte Programações baseadas em hora para tarefas e crawlers.

Erro: já existe um acionador com o mesmo nome

O AWS Glue retorna o erro “Trigger with name 'foo_starting_trigger' already submitted with different configuration” (Acionador com o nome ‘foo_starting_trigger’ já enviado com configuração diferente).

Um blueprint não exige que você defina acionadores no script de layout para a criação do fluxo de trabalho. A criação do acionador é gerenciada pela biblioteca do blueprint com base nas dependências definidas entre duas ações.

A nomeação para os acionadores é a seguinte:

- Para o acionador inicial no fluxo de trabalho, a nomeação é <workflow_name>_starting_trigger.
- Para um nó (trabalho/crawler) no fluxo de trabalho que depende da conclusão de um ou vários nós upstream, o AWS Glue define um acionador com o nome <workflow_name>_<node_name>_trigger

Este erro significa que já existe um acionador com o mesmo nome. Você pode excluir o acionador existente e executar novamente a criação do fluxo de trabalho.

> **Note**
>
> Excluir um fluxo de trabalho não exclui os nós dentro do fluxo de trabalho. É possível que, embora o fluxo de trabalho seja excluído, os acionadores sejam deixados para trás. Devido a isso, você pode não receber um erro “workflow already exists” (o fluxo de trabalho já existe), mas você pode receber um erro “trigger already exists” (o acionador já existe) em um caso em que você cria um fluxo de trabalho, exclui e, em seguida, tenta recriá-lo com o mesmo nome do mesmo blueprint.

Erro: workflow with name: foo already exists (o fluxo de trabalho com nome: foo já existe).

O nome do fluxo de trabalho deve ser exclusivo. Tente com outro nome.

Erro: módulo não encontrado no caminho layoutGenerator especificado

O AWS Glue retorna o erro “Unknown error executing layout generator function ModuleNotFoundError: No module named ‘crawl_s3_locations’” (Erro desconhecido ao executar a função do gerador de layout ModuleNotFounderror: nenhum módulo chamado “crawl_s3_locations”).

```
layoutGenerator": "crawl_s3_locations.layout.generate_layout"
```

Por exemplo, se você tiver o caminho layoutGenerator acima, quando descompactar o arquivo de blueprint, ele precisará se parecer com o seguinte:

```
$ 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
```

Ao descompactar o arquivo, se o arquivo de blueprint se parecer com o seguinte, então você poderá obter o erro acima.

```
$ unzip crawl_s3_locations.zip
Archive:  crawl_s3_locations.zip
  inflating: blueprint.cfg
  inflating: layout.py
  inflating: README.md
```

Você pode ver que não há nenhuma pasta chamada `crawl_s3_locations` e, quando o caminho `layoutGenerator` refere-se ao arquivo de layout por meio do módulo `crawl_s3_locations`, você pode obter o erro acima.

Erro: erro de validação no campo Connections (Conexões)

O AWS Glue retorna o erro “Unknown error executing layout generator function TypeError: Value ['foo'] for key Connections should be of type <class 'dict'>!” (Erro desconhecido ao executar a função do gerador de layout TypeError: Valor ['foo'] para chave Conexões deve ser do tipo <class 'dict'>!).

Este é um erro de validação. O campo `Connections` na classe `Job` está esperando um dicionário e, em vez disso, uma lista de valores é fornecida, causando o erro.

```
User input was list of values
```

```
Connections= ['string']

Should be a dict like the following
Connections*=*{'Connections': ['string']}
```

Para evitar esses erros de runtime ao criar um fluxo de trabalho de um esquema, você pode validar as definições de fluxo de trabalho, trabalho e crawler, conforme descrito em Testar um esquema.

Consulte a sintaxe na Referência de classes de esquema do AWS Glue para definir o trabalho, crawler e fluxo de trabalho do AWS Glue no script de layout.

Permissões para pessoas e funções de esquemas do AWS Glue

A seguir estão as pessoas típicas e as políticas de permissões do AWS Identity and Access Management (IAM) sugeridas para pessoas e funções para os esquemas do AWS Glue.

Tópicos

- Pessoas de esquemas
- Permissões para pessoas de esquemas
- Permissões para perfis de esquema

Pessoas de esquemas

A seguir estão as pessoas normalmente envolvidas no ciclo de vida dos esquemas do AWS Glue.

Pessoa	Descrição
Desenvolvedor do AWS Glue	Desenvolve, testa e publica blueprints.
Administrador da AWS Glue	Regista, mantém e concede permissões em blueprints.
Analista de dados	Executa blueprints para criar fluxos de trabalho.

Para ter mais informações, consulte the section called “Visão geral dos esquemas”.

Permissões para pessoas de esquemas

A seguir estão as permissões sugeridas para cada pessoa do blueprint.

Permissões de desenvolvedor do AWS Glue para esquemas

O desenvolvedor do AWS Glue deve ter permissões de gravação no bucket do Amazon S3 usado para publicar o esquema. Muitas vezes, o desenvolvedor registra o blueprint depois de carregá-lo. Nesse caso, o desenvolvedor precisa das permissões listadas em the section called “Permissões de administrador do AWS Glue para esquemas”. Além disso, se o desenvolvedor desejar testar o blueprint após seu registro, ele ou ela também precisará das permissões listadas em the section called “Permissões de analista de dados para esquemas”.

Permissões de administrador do AWS Glue para esquemas

A política a seguir concede permissões para registrar, visualizar e manter esquemas do AWS Glue.

> **Important**
>
> Na política a seguir, substitua *<s3-bucket-name>* e *<prefix>* com o caminho do Amazon S3 dos arquivos ZIP de blueprint carregados a serem registrados.

```
{
    "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>/*"
        }
```

```
    ]
}
```

Permissões de analista de dados para esquemas

A política a seguir concede permissões para executar blueprints e exibir o fluxo de trabalho resultante e seus componentes. Ela também concede `PassRole` para a função que o AWS Glue assume para criar o fluxo de trabalho resultante e seus componentes.

A política concede permissões em qualquer recurso. Se você deseja configurar o acesso minucioso a blueprints individuais, use o seguinte formato para ARNs de blueprint:

```
arn:aws:glue:<region>:<account-id>:blueprint/<blueprint-name>
```

> **Important**
>
> Na política a seguir, substitua *<account-id>* por uma conta da AWS válida e substitua *<role-name>* pelo nome da função usada para executar um blueprint. Consulte the section called “Permissões para perfis de esquema” para obter as permissões que essa função requer.

```
{
    "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>"
        }
    ]
}
```

Permissões para perfis de esquema

A seguir estão as permissões sugeridas para a função do IAM usada para criar um fluxo de trabalho a partir de um blueprint. A função deve ter uma relação de confiança com o `glue.amazonaws.com`.

> **Important**
>
> Na política a seguir, substitua *<account-id>* por uma conta da AWS válida e substitua *<role-name>* pelo nome da função.

```
{
    "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**
>
> Se os trabalhos e crawlers no fluxo de trabalho assumirem uma função diferente dessa, tal política deverá incluir a permissão `iam:PassRole` nessa outra função, em vez de na função de blueprint.

Desenvolver esquemas no AWS Glue

Sua organização pode ter um conjunto de casos de uso de ETL semelhantes que poderiam se beneficiar da capacidade de parametrizar um único fluxo de trabalho para lidar com todos eles. Para atender a essa necessidade, o AWS Glue permite que você defina esquemas, que você pode usar para gerar fluxos de trabalho. Um blueprint aceita parâmetros, de modo que, a partir de um único blueprint, um analista de dados possa criar fluxos de trabalho diferentes para lidar com casos de uso de ETL semelhantes. Depois de criar um blueprint, é possível reutilizá-lo em departamentos, equipes e projetos diferentes.

Tópicos

- Atualizar um esquema no AWS Glue
- Criar um fluxo de trabalho com base em um esquema no AWS Glue
- Visualizar execuções de esquemas no AWS Glue

Visão geral dos esquemas no AWS Glue

> **Note**
>
> No momento, o recurso de esquemas não está disponível no console do AWS Glue nas seguintes regiões: Ásia-Pacífico (Jacarta) e Oriente Médio (EAU).

Os esquemas do AWS Glue fornecem uma maneira de criar e compartilhar fluxos de trabalho do AWS Glue. Quando há um processo de ETL complexo que poderia ser usado para casos de uso semelhantes, em vez de criar um fluxo de trabalho do AWS Glue para cada caso de uso, você pode criar um único esquema.

O blueprint especifica os trabalhos e crawlers a serem incluídos em um fluxo de trabalho e especifica os parâmetros que o usuário do fluxo fornece ao executar o blueprint para criar um fluxo de trabalho. O uso de parâmetros permite que um único blueprint gere fluxos de trabalho para os vários casos de uso semelhantes. Para obter mais informações sobre fluxos de trabalho, consulte Visão geral de fluxos de trabalho no AWS Glue.

Veja a seguir exemplos de casos de uso de blueprints:

- Você deseja particionar um conjunto de dados existente. Os parâmetros de entrada para o blueprint são caminhos de fonte e de destino do Amazon Simple Storage Service (Amazon S3) e uma lista de colunas de partição.
- Você deseja fazer um snapshot de uma tabela do Amazon DynamoDB em um armazenamento de dados SQL, como o Amazon RedShift. Os parâmetros de entrada para o esquema são o nome da tabela do DynamoDB e uma conexão do AWS Glue, que designa um cluster do Amazon RedShift e um banco de dados de destino.
- Você deseja converter dados em CSV em vários caminhos do Amazon S3 para Parquet. Você deseja que o fluxo de trabalho do AWS Glue inclua um crawler e um trabalho separados para cada caminho. Os parâmetros de entrada são o banco de dados de destino no catálogo de dados do AWS Glue e uma lista delimitada por vírgulas de caminhos do Amazon S3. Observe que nesse caso, o número de crawlers e trabalhos criados pelo fluxo de trabalho é variável.

Componentes do esquema

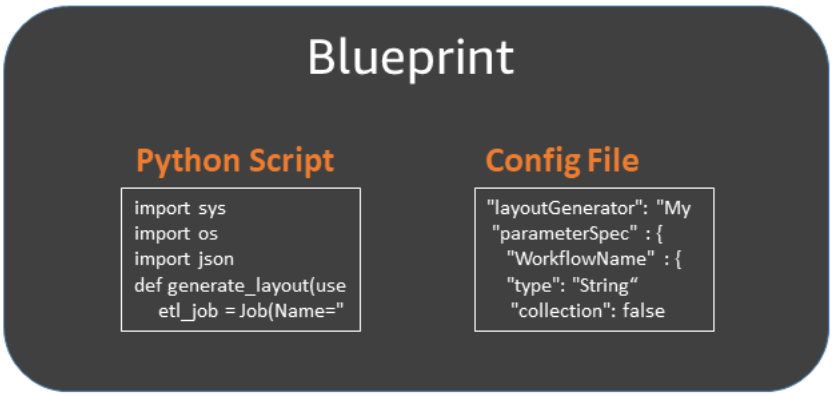
Um blueprint é um arquivo ZIP que contém os seguintes componentes:

- Um script gerador de layout Python

  Contém uma função que especifica o layout do fluxo de trabalho. Os crawlers e trabalhos a serem criados para o fluxo de trabalho, as propriedades do trabalho e do crawler e as dependências entre eles. A função aceita parâmetros de esquema e retorna uma estrutura de fluxo de trabalho (objeto JSON) que o AWS Glue usa para gerar o fluxo de trabalho. Como você usa um script Python para gerar o fluxo de trabalho, pode adicionar sua própria lógica adequada a seus casos de uso.
- Um arquivo de configuração

  Especifica o nome totalmente qualificado da função Python que gera o layout do fluxo de trabalho. Especifica também os nomes, tipos de dados e outras propriedades de todos os parâmetros do blueprint usados pelo script.
- (Opcional) scripts de ETL e arquivos complementares

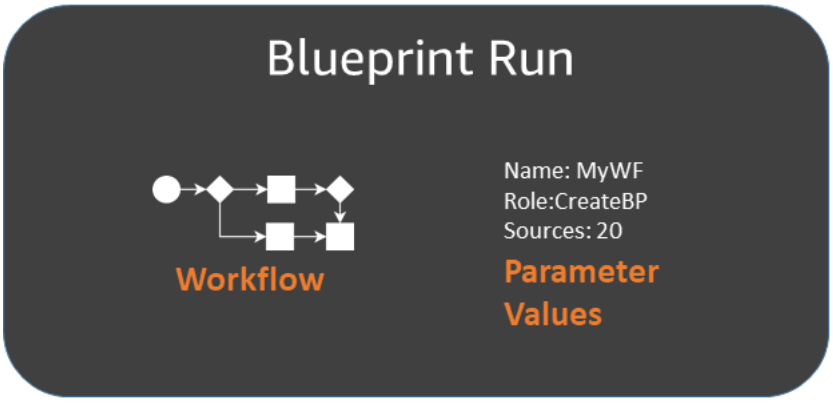
  Como um caso de uso avançado, você pode parametrizar a localização dos scripts de ETL que seus trabalhos usam. Você pode incluir arquivos de script de trabalho no arquivo ZIP e especificar um parâmetro de blueprint para um local do Amazon S3 para o qual os scripts devem ser copiados. O script gerador de layout pode copiar os scripts de ETL para o local designado e especificá-lo como a propriedade de local do script do trabalho. Você também pode incluir quaisquer bibliotecas ou outros arquivos complementares, desde que seu script os manipule.



Execuções de blueprint

Quando você cria um fluxo de trabalho a partir de um esquema, o AWS Glue executa o esquema, que inicia um processo assíncrono para criar o fluxo de trabalho e os trabalhos, crawlers e

acionadores que o fluxo de trabalho encapsula. O AWS Glue usa a execução do esquema para orquestrar a criação do fluxo de trabalho e seus componentes. Você visualizar o status do processo de criação exibindo o status de execução do blueprint. A execução do blueprint também armazena os valores que você forneceu para os parâmetros do blueprint.



Você pode exibir execuções de esquemas usando o console do AWS Glue ou a AWS Command Line Interface (AWS CLI). Ao exibir ou solucionar problemas de um fluxo de trabalho, você sempre pode retornar à execução do blueprint para exibir os valores de parâmetro do blueprint usados para criar o fluxo de trabalho.

Ciclo de vida de um esquema

Os esquemas são desenvolvidos, testados e registrados com o AWS Glue e executados para criar fluxos de trabalho. Normalmente, há três pessoas envolvidas no ciclo de vida do blueprint.

Pessoa	Tarefas
Desenvolvedor do AWS Glue	• Escreve o script de layout do fluxo de trabalho e cria o arquivo de configuração. • Testa o esquema localmente usando bibliotecas fornecidas pelo serviço do AWS Glue. • Cria um arquivo ZIP do script, do arquivo de configura ção e dos arquivos complementares e o publica em um local no Amazon S3. • Adiciona uma política de bucket ao bucket do Amazon S3 que concede permissões de leitura em objetos de bucket para a conta da AWS do administrador do AWS Glue.

Pessoa	Tarefas
	• Concede permissões de leitura do IAM no arquivo ZIP no Amazon S3 para o administrador do AWS Glue.
Administrador da AWS Glue	• Registra o esquema com o AWS Glue. O AWS Glue faz uma cópia do arquivo ZIP em um local reservado do Amazon S3. • Concede permissões do IAM no blueprint aos analistas de dados.
Analista de dados	• Executa o blueprint para criar um fluxo de trabalho e fornece valores de parâmetro do blueprint. Verifica o status de execução do blueprint para garantir que o fluxo de trabalho e seus componentes tenham sido gerados com êxito. • Executa e soluciona problemas do fluxo de trabalho. Antes de executar o fluxo de trabalho, pode verificar o fluxo de trabalho exibindo o gráfico de design dele, no console do AWS Glue.

Consulte também

- Desenvolver esquemas no AWS Glue
- Criar um fluxo de trabalho com base em um esquema no AWS Glue
- Permissões para pessoas e funções de esquemas do AWS Glue

Desenvolver esquemas no AWS Glue

Como um desenvolvedor do AWS Glue, você pode criar e publicar esquemas que os analistas de dados podem usar para gerar fluxos de trabalho.

Tópicos

- Visão geral do desenvolvimento de esquemas
- Pré-requisitos para desenvolvimento de esquemas

- Gravar o código do esquema
- Projeto de esquema de exemplo
- Testar um esquema
- Publicar um esquema
- Referência de classes de esquema do AWS Glue
- Esquemas de exemplo

Consulte também

- Visão geral dos esquemas no AWS Glue

Visão geral do desenvolvimento de esquemas

A primeira etapa em seu processo de desenvolvimento é identificar um caso de uso comum que se beneficiaria de um blueprint. Um caso de uso típico envolve um problema de ETL recorrente que você acredita que deve ser resolvido de forma geral. Em seguida, projete um blueprint que implante o caso de uso generalizado e defina os parâmetros de entrada do blueprint que juntos podem definir um caso de uso específico a partir do caso de uso generalizado.

Um blueprint consiste em um projeto que contém um arquivo de configuração de parâmetros do blueprint e um script que define o layout do fluxo de trabalho a ser gerado. O layout define os trabalhos e crawlers (ou entidades na terminologia do script de blueprint) a serem criados.

Você não especifica diretamente nenhum acionador no script de layout. Em vez disso, você escreve um código para especificar as dependências entre os trabalhos e os crawlers criados pelo script. O AWS Glue gera os acionadores com base em suas especificações de dependências. A saída do script de layout é um objeto de fluxo de trabalho que contém especificações para todas as entidades de fluxo de trabalho.

Você constrói seu objeto de fluxo de trabalho usando as seguintes bibliotecas de esquema do AWS Glue:

- `awsglue.blueprint.base_resource`: uma biblioteca de recursos básicos usados pelas bibliotecas.
- `awsglue.blueprint.workflow`: uma biblioteca para definir uma classe `Workflow`.

- `awsglue.blueprint.job`: uma biblioteca para definir uma classe `Job`.
- `awsglue.blueprint.crawler`: uma biblioteca para definir uma classe `Crawler`.

As únicas outras bibliotecas que são suportadas para geração de layout são aquelas que estão disponíveis para o shell do Python.

Antes de publicar seu blueprint, você pode usar métodos definidos nas bibliotecas de blueprint para testá-lo localmente.

Quando estiver tudo pronto para você disponibilizar o blueprint aos analistas de dados, você empacota o script, o arquivo de configuração de parâmetros e quaisquer arquivos complementares, como scripts e bibliotecas adicionais, em um único ativo implantável. Em seguida, carrega o ativo no Amazon S3 e pede a um administrador para registrá-lo no AWS Glue.

Para obter mais informações sobre amostras de projetos de blueprint, consulte Projeto de esquema de exemplo e Esquemas de exemplo.

Pré-requisitos para desenvolvimento de esquemas

Para desenvolver esquemas, você deve ter familiaridade com o uso do AWS Glue e com a criação de scripts para trabalhos de ETL do Apache Spark ou trabalhos de shell do Python. Além disso, é necessário concluir as seguintes tarefas de configuração.

- Baixe quatro bibliotecas Python da AWS para usar em seus scripts de layout de blueprint.
- Configure os AWS SDKs.
- Configure o AWS CLI.

Baixar bibliotecas Python

Baixe as seguintes bibliotecas do GitHub e instale-as em seu projeto:

- 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

Configurar o AWS Java SDK

Para o AWS Java SDK, você deve adicionar um arquivo `jar` que inclui a API para blueprints.

1. Se você ainda não fez isso, configure o AWS SDK for Java.
   - Para Java 1.x, siga as instruções em Configurar o AWS SDK for Java no Guia do desenvolvedor do AWS SDK for Java.
   - Para Java 2.x, siga as instruções em Configurar o AWS SDK for Java 2.x no Guia do desenvolvedor do AWS SDK for Java 2.x.
2. Baixe o arquivo de cliente `jar` que tem acesso às APIs para blueprints.
   - 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. Adicione o cliente `jar` na frente do classpath Java para substituir o cliente do AWS Glue fornecido peloAWS Java SDK.

```
export CLASSPATH=<path-to-preview-client-jar>:$CLASSPATH
```

4. (Opcional) teste o SDK com a seguinte aplicação Java. A aplicação deve exibir uma lista vazia.

   Substitua `accessKey` e `secretKey` com suas credenciais e `us-east-1` com a sua região.

```
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));
    }
}
```

Configurar o AWS Python SDK

As seguintes etapas supõem que você tenha o Python versão 2.7 ou posterior, ou a versão 3.6 ou posterior, instalada no computador.

1. Baixe o seguinte arquivo wheel do boto3. Se solicitado a abrir ou salvar, salve o arquivo. s3://awsglue-custom-blueprints-preview-artifacts/aws-python-sdk-preview/boto3-1.17.31-py2.py3-none-any.whl
2. Baixe o seguinte arquivo wheel do botocore: s3://awsglue-custom-blueprints-preview-artifacts/aws-python-sdk-preview/botocore-1.20.31-py2.py3-none-any.whl
3. Verifique a versão do Python.

   ```
   python --version
   ```
4. Dependendo da sua versão do Python, insira os seguintes comandos (para Linux):

   - Para Python 2.7 ou posterior.

     ```
     python3 -m pip install --user virtualenv
     source env/bin/activate
     ```
   - Para Python 3.6 ou posterior.

     ```
     python3 -m venv python-sdk-test
     source python-sdk-test/bin/activate
     ```
5. Instale o arquivo wheel do botocore.

   ```
   python3 -m pip install <download-directory>/botocore-1.20.31-py2.py3-none-any.whl
   ```
6. Instale o arquivo wheel do boto3.

   ```
   python3 -m pip install <download-directory>/boto3-1.17.31-py2.py3-none-any.whl
   ```

7. Configure suas credenciais e a região padrão nos arquivos `~/.aws/credentials` e `~/.aws/config`. Para obter mais informações, consulte Configuração da AWS CLI no Guia do usuário da AWS Command Line Interface.
8. (Opcional) teste a configuração. Os comandos a seguir devem retornar uma lista vazia.

   Substitua `us-east-1` pela sua região.

   ```
   $ python
   >>> import boto3
   >>> glue = boto3.client('glue', 'us-east-1')
   >>> glue.list_blueprints()
   ```

Configurar a pré-visualização da AWS CLI

1. Se você ainda o fez, instale e/ou atualize a AWS Command Line Interface (AWS CLI) em seu computador. A maneira mais fácil de fazer isso é com o `pip`, o utilitário de instalação do Python:

   ```
   pip install awscli --upgrade --user
   ```

   Você pode encontrar instruções completas de instalação da AWS CLI aqui: Instalar a AWS Command Line Interface.
2. Baixe o arquivo wheel da AWS CLI em: s3://awsglue-custom-blueprints-preview-artifacts/awscli-preview-build/awscli-1.19.31-py2.py3-none-any.whl
3. Instale o arquivo wheel da AWS CLI.

   ```
   python3 -m pip install awscli-1.19.31-py2.py3-none-any.whl
   ```
4. Execute o comando `aws configure`. Configure as credenciais da AWS (incluindo chave de acesso e chave secreta) e a região da AWS. Você pode encontrar informações sobre a configuração da AWS CLI aqui: Configurar a AWS CLI.
5. Teste a AWS CLI. O comando a seguir deve retornar uma lista vazia.

   Substitua `us-east-1` pela sua região.

   ```
   aws glue list-blueprints --region us-east-1
   ```

Gravar o código do esquema

Cada projeto de blueprint criado deve conter, no mínimo, os seguintes arquivos:

- Um script de layout Python que define o fluxo de trabalho. O script contém uma função que define as entidades (trabalhos e crawlers) em um fluxo de trabalho e as dependências entre elas.
- Um arquivo de configuração, `blueprint.cfg`, que define:
  - O caminho completo da função de definição de layout de fluxo de trabalho.
  - Os parâmetros que o blueprint aceita.

Tópicos

Criar o script de layout do esquema

O script de layout do blueprint deve incluir uma função que gera as entidades em seu fluxo de trabalho. Você pode atribuir o nome que quiser a esta função. O AWS Glue usa o arquivo de configuração para determinar o nome totalmente qualificado da função.

Sua função de layout faz o seguinte:

- (Opcional) instancia a classe `Job` para criar objetos `Job` e transmite argumentos, como `Command` e `Role`. Essas são as propriedades de trabalho que você especificaria se estivesse criando o trabalho usando o console ou a API do AWS Glue.
- (Opcional) instancia a classe `Crawler` para criar objetos `Crawler` e transmite argumentos de nome, função e destino.
- Para indicar dependências entre os objetos (entidades de fluxo de trabalho), transmite os argumentos adicionais `DependsOn` e `WaitForDependencies` para `Job()` e `Crawler()`. Esses argumentos são explicados posteriormente nesta seção.
- Instancia a classe `Workflow` para criar o objeto de fluxo de trabalho que é retornado para o AWS Glue, transmitindo um argumento `Name`, um argumento `Entities` e um argumento opcional `OnSchedule`. O argumento `Entities` especifica todos os trabalhos e crawlers a serem incluídos no fluxo de trabalho. Para ver como construir um objeto `Entities`, consulte o projeto de exemplo mais adiante nesta seção.

- Retorna um objeto `Workflow`.

Para obter definições das classes `Job`, `Crawler` e `Workflow`, consulte Referência de classes de esquema do AWS Glue.

A função de layout aceita os seguintes argumentos:

Argumento	Descrição
`user_params`	Dicionário Python de nomes e valores de parâmetros do blueprint. Para ter mais informações, consulte Especificar parâmetros de esquema.
`system_params`	Dicionário Python contendo duas propriedades: `region` e `accountId` .

Aqui está um exemplo de script gerador de layout em um arquivo chamado `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
```

O script de exemplo importa as bibliotecas de blueprint necessárias e inclui uma função `generate_layout` que gera um fluxo de trabalho com dois trabalhos. Esse é um script muito simples. Um script mais complexo poderia empregar lógica e parâmetros adicionais para gerar um fluxo de trabalho com muitos trabalhos e crawlers, ou até mesmo um número variável de trabalhos e crawlers.

Usar o argumento DependsOn

O argumento `DependsOn` é uma representação de dicionário de uma dependência que esta entidade tem em outras entidades dentro do fluxo de trabalho. Ele tem o seguinte formato:

```
DependsOn = {dependency1 : state, dependency2 : state, ...}
```

As chaves nesse dicionário representam a referência ao objeto, e não o nome da entidade, enquanto os valores são strings que correspondem ao estado a ser vigiado. O AWS Glue infere os acionadores adequados. Para obter os estados válidos, consulte Estrutura de condição.

Por exemplo, um trabalho pode depender da conclusão bem-sucedida de um crawler. Se você definir um objeto de crawler chamado `crawler2` como segue:

```
crawler2 = Crawler(Name="my_crawler", ...)
```

Em seguida, um objeto dependendo de `crawler2` incluiria um argumento construtor, como:

```
DependsOn = {crawler2 : "SUCCEEDED"}
```

Por exemplo:

```
job1 = Job(Name="Job1", ..., DependsOn = {crawler2 : "SUCCEEDED", ...})
```

Se `DependsOn` for omitido para uma entidade, essa entidade dependerá do acionador de início do fluxo de trabalho.

Usar o argumento WaitForDependencies

O argumento `WaitForDependencies` define se um trabalho ou entidade de crawler deve esperar até que todas as entidades das quais depende sejam concluídas ou até que qualquer uma seja concluída.

Os valores permitidos são “`AND`” ou “`ANY`”.

Usar o argumento OnSchedule

O argumento `OnSchedule` para o construtor de classe `Workflow` é uma expressão `cron` que determina a definição do acionador inicial para um fluxo de trabalho.

Se esse argumento for especificado, o AWS Glue criará um acionador de programação com a programação correspondente. Se ele não for especificado, o acionador de início do fluxo de trabalho será um acionador sob demanda.

Criar o arquivo de configuração

O arquivo de configuração do blueprint é um arquivo obrigatório que define o ponto de entrada do script para gerar o fluxo de trabalho e os parâmetros que o blueprint aceita. O deve ser nomeado `blueprint.cfg`.

Veja a seguir um arquivo de configuração de exemplo.

```
{
    "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
     }
    }
}
```

A propriedade `layoutGenerator` especifica o nome totalmente qualificado da função no script que gera o layout.

A propriedade `parameterSpec` especifica os parâmetros que esse blueprint aceita. Para ter mais informações, consulte Especificar parâmetros de esquema.

> **Important**
>
> Seu arquivo de configuração deve incluir o nome do fluxo de trabalho como um parâmetro de blueprint ou você deve gerar um nome de fluxo de trabalho exclusivo no script de layout.

Especificar parâmetros de esquema

O arquivo de configuração contém especificações de parâmetro do blueprint em um objeto JSON `parameterSpec`. `parameterSpec` contém um ou mais objetos de parâmetro.

```
"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>": {
       ...
    }
  }
```

A seguir estão as regras para codificar cada objeto de parâmetro:

- O nome e o `type` do parâmetro são obrigatórios. Todas as outras propriedades são opcionais.
- Se você especificar a propriedade `defaultValue`, o parâmetro será opcional. Caso contrário, o parâmetro é obrigatório e o analista de dados que cria um fluxo de trabalho a partir do blueprint deve fornecer um valor para ele.
- Se você definir a propriedade `collection` como `true`, o parâmetro poderá ter uma coleção de valores. Coleções podem ser de qualquer tipo de dados.
- Se você especificar `allowedValues`, o console do AWS Glue exibirá uma lista suspensa de valores para o analista de dados escolher ao criar um fluxo de trabalho a partir do esquema.

Os seguintes valores são permitidos para o `type`:

Tipos de dados de parâmetro	Observações
`String`	-
`Integer`	-
`Double`	-
`Boolean`	Os possíveis valores são `true` e `false`. Gera uma caixa de seleção na página Create a workflow from <esquema> (Criar um fluxo de trabalho a partir de <esquema>) no console do AWS Glue.
`S3Uri`	Caminho completo do Amazon S3, começando com `s3://`. Gera um campo de texto e o botão Browse (Navegar) na página Create a workflow from <blueprint> (Criar um fluxo de trabalho a partir de <blueprint>).
`S3Bucket`	Somente nome do bucket do Amazon S3. Gera um seletor de bucket na página Create a workflow from <blueprint> (Criar um fluxo de trabalho a partir de <blueprint>).
`IAMRoleArn`	O nome do recurso da Amazon (ARN) da função do AWS Identity and Access Management (IAM). Gera um seletor de função na página Create a workflow from <blueprint> (Criar um fluxo de trabalho a partir de <blueprint>).

Tipos de dados de parâmetro	Observações
`IAMRoleName`	Nome de uma função do IAM. Gera um seletor de função na página Create a workflow from <blueprint> (Criar um fluxo de trabalho a partir de <blueprint>).

Projeto de esquema de exemplo

Uma conversão de formato de dados é um caso de uso frequente de extração, transformação e carregamento (ETL). Em workloads analíticas típicas, formatos de arquivo baseados em colunas, como Parquet ou ORC, são preferidos em relação a formatos de texto, como CSV ou JSON. Esse blueprint de exemplo permite converter dados de CSV/JSON/etc. em Parquet para arquivos no Amazon S3.

Esse blueprint obtém uma lista de caminhos do S3 definidos por um parâmetro de blueprint, converte os dados no formato Parquet e os grava no local do S3 especificado por outro parâmetro de blueprint. O script de layout cria um crawler e um trabalho para cada caminho. O script de layout também carrega o script de ETL no `Conversion.py` para um bucket do S3 especificado por outro parâmetro do blueprint. Em seguida, o script de layout especifica o script carregado como o script de ETL para cada trabalho. O arquivo ZIP do projeto contém o script de layout, o script de ETL e o arquivo de configuração do blueprint.

Para obter mais informações sobre exemplos de projetos de blueprint, consulte Esquemas de exemplo.

A seguir está o script de layout, no arquivo `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
```

A seguir está o arquivo de configuração do blueprint correspondente `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."
        }
    }
}
```

O script a seguir no arquivo `Conversion.py` é o script de ETL carregado. Observe que ele preserva o esquema de particionamento durante a conversão.

```
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**
>
> Somente dois caminhos do Amazon S3 podem ser fornecidos como uma entrada para o blueprint de exemplo. Isso porque os acionadores do AWS Glue podem invocar apenas duas ações de crawler.

Testar um esquema

Enquanto você desenvolve seu código, você deve executar testes locais para verificar se o layout do fluxo de trabalho está correto.

Testes locais não geram trabalhos, crawlers ou acionadores do AWS Glue. Em vez disso, execute o script de layout localmente e use os métodos `to_json()` e `validate()` para imprimir objetos e encontrar erros. Esses métodos estão disponíveis em todas as três classes definidas nas bibliotecas.

Há duas maneiras de lidar com os argumentos `user_params` e `system_params` que o AWS Glue transmite para sua função de layout. Seu código de banco de teste pode criar um dicionário de valores de parâmetro de blueprint de exemplo e transmiti-lo para a função de layout como o argumento `user_params`. Ou você pode remover as referências a `user_params` e substituí-las por strings em código fixo.

Se o seu código usar as propriedades `region` e `accountId` no argumento `system_params`, você pode transmitir seu próprio dicionário para `system_params`.

Para testar um blueprint

1. Inicie um interpretador Python em um diretório com as bibliotecas ou carregue os arquivos de blueprint e as bibliotecas fornecidas em seu ambiente de desenvolvimento integrado (IDE) preferido.
2. Certifique-se de que o código importe as bibliotecas fornecidas.
3. Adicione código à sua função de layout para chamar `validate()` ou `to_json()` em qualquer entidade ou no objeto `Workflow`. Por exemplo, se o seu código criar um objeto `Crawler` chamado `mycrawler`, é possível chamar `validate()` como a seguir.

   ```
   mycrawler.validate()
   ```

   Você pode imprimir `mycrawler` como a seguir:

   ```
   print(mycrawler.to_json())
   ```

   Se você chamar `to_json` em um objeto, não há necessidade de também chamar `validate()`, pois `to_json()` chama `validate()`.

   É mais útil chamar esses métodos no objeto de fluxo de trabalho. Supondo que seu script nomeie o objeto de fluxo de trabalho `my_workflow`, valide e imprima o objeto de fluxo de trabalho da seguinte forma:

   ```
   print(my_workflow.to_json())
   ```

   Para obter mais informações sobre `to_json()` e `validate()`, consulte Classe Methods.

   Você também pode importar `pprint` e formatar o estilo do objeto do fluxo de trabalho, conforme mostrado no exemplo mais adiante nesta seção.

4. Execute o código, corrija erros e, finalmente, remova quaisquer chamadas para `validate()` ou `to_json()`.

Example

O exemplo a seguir mostra como construir um dicionário de parâmetros de blueprint de exemplo e transmiti-lo como o argumento `user_params` para a função de layout `generate_compaction_workflow`. Ele também mostra como formatar o estilo do objeto de fluxo de trabalho gerado.

```
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)
```

Publicar um esquema

Depois de desenvolver um blueprint, você deve carregá-lo no Amazon S3. Você deve ter permissões de gravação no bucket do Amazon S3 usado para publicar o blueprint. Você também deve se certificar de que o administrador do AWS Glue, que registrará o esquema, tenha acesso de leitura ao bucket do Amazon S3. Sobre políticas de permissões do AWS Identity and Access Management (IAM) sugeridas para pessoas e funções para esquemas do AWS Glue, consulte Permissões para pessoas e funções de esquemas do AWS Glue.

Para publicar um blueprint

1. Crie os scripts, os recursos e o arquivo de configuração do blueprint necessários.
2. Adicione todos os arquivos a um arquivo ZIP e carregue-o no Amazon S3. Use um bucket do S3 que esteja na mesma região na qual os usuários se registrarão e executarão o blueprint.

   Você pode criar um arquivo ZIP a partir da linha de comando usando o seguinte comando.

   ```
   zip -r folder.zip folder
   ```

3. Adicione uma política de bucket que conceda permissões de leitura à conta da AWS desejada. Veja a seguir um exemplo de política.

   ```
   {
   ```

```
    "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 a permissão do IAM `s3:GetObject` no bucket do Amazon S3 para o administrador do AWS Glue ou para quem quer que esteja registrando esquemas. Para obter uma política de exemplo a conceder aos administradores, consulte Permissões de administrador do AWS Glue para esquemas.

Depois de concluir o teste local do seu esquema, você também pode querer testar um esquema no AWS Glue. Para testar um esquema no AWS Glue, ele deve ser registrado. Você pode limitar quem vê o blueprint registrado usando a autorização do IAM ou usando contas de teste separadas.

> Consulte também:
>
> - Registrar um esquema no AWS Glue

Referência de classes de esquema do AWS Glue

As bibliotecas para esquemas do AWS Glue definem três classes que você pode usar no script de layout do fluxo de trabalho: `Job`, `Crawler` e `Workflow`.

Tópicos

- Classe Job
- Classe Crawler
- Classe Workflow
- Classe Methods

Classe Job

A classe `Job` representa um trabalho de ETL do AWS Glue.

Argumentos construtores obrigatórios

Veja os argumentos construtores obrigatórios para a classe `Job`.

Nome do argumento	Tipo	Descrição
`Name`	`str`	O nome a ser atribuído ao trabalho. O AWS Glue adiciona um sufixo gerado aleatoriamente ao nome para distinguir o trabalho daqueles criados por outras execuções de esquema.
`Role`	`str`	O nome do recurso da Amazon (ARN) da função que o trabalho deve assumir ao executar.
`Command`	`dict`	Comando Job (Trabalho), conforme especificado em JobCommand estrutura na documentação da API.

Argumentos construtores opcionais

Veja os argumentos construtores opcionais para a classe `Job`.

Nome do argumento	Tipo	Descrição
`DependsOn`	`dict`	Lista de entidades de fluxo de trabalho das quais o trabalho depende. Para ter mais informações, consulte Usar o argumento DependsOn.
`WaitForDependencies`	`str`	Indica se o trabalho deve esperar até que todas as entidades das quais depende sejam concluídas antes de executar ou até que qualquer uma seja concluída. Para ter mais informações, consulte Usar o argumento

Nome do argumento	Tipo	Descrição
		WaitForDependencies. Omita, se o trabalho depender de apenas uma entidade.
(Propriedades do trabalho)	-	Qualquer uma das propriedades do trabalho listadas em Estrutura Job na documentação da API do AWS Glue (exceto `CreatedOn` e `LastModifiedOn`).

Classe Crawler

A classe `Crawler` representa um crawler do AWS Glue.

Argumentos construtores obrigatórios

Veja os argumentos construtores obrigatórios para a classe `Crawler`.

Nome do argumento	Tipo	Descrição
`Name`	`str`	O nome a ser atribuído ao crawler. O AWS Glue adiciona um sufixo gerado aleatoriamente ao nome para distinguir o trabalho daqueles criados por outras execuções de esquema.
`Role`	`str`	ARN da função que o crawler deve assumir ao executar.
`Targets`	`dict`	Coleção de destinos a serem rastreados. Os argumentos construtores da classe `Targets` são definidos em CrawlerTargets estrutura na documentação da API. Todos os argumentos construtores de `Targets` são opcionais, mas você deve informar pelo menos um.

Argumentos construtores opcionais

Veja os argumentos construtores opcionais para a classe `Crawler`.

Nome do argumento	Tipo	Descrição
`DependsOn`	`dict`	Lista de entidades de fluxo de trabalho das quais o crawler depende. Para ter mais informações, consulte Usar o argumento DependsOn.
`WaitForDependencies`	`str`	Indica se o crawler deve esperar até que todas as entidades das quais depende sejam concluídas antes de executar ou até que qualquer uma seja concluída. Para ter mais informações, consulte Usar o argumento WaitForDependencies. Omita, se o crawler depender de apenas uma entidade.
(Propriedades do crawler)	-	Qualquer uma das propriedades do crawler listadas em Estrutura Crawler na documentação da API do AWS Glue, com as seguintes exceções: • `State` • `CrawlElapsedTime` • `CreationTime` • `LastUpdated` • `LastCrawl` • `Version`

Classe Workflow

A classe `Workflow` representa um fluxo de trabalho do AWS Glue. O script de layout de fluxo de trabalho retorna um objeto `Workflow`. AWS Glue cria um fluxo de trabalho com base nesse objeto.

Argumentos construtores obrigatórios

Veja os argumentos construtores obrigatórios para a classe `Workflow`.

Nome do argumento	Tipo	Descrição
`Name`	`str`	O nome a ser atribuído ao fluxo de trabalho.
`Entities`	`Entities`	Uma coleção de entidades (trabalhos e crawlers) a serem incluídas no fluxo de trabalho. O construtor da classe `Entities` aceita um argumento `Jobs`, que é uma lista de objetos `Job`, e um argumento `Crawlers`, que é uma lista de objetos `Crawler`.

Argumentos construtores opcionais

Veja os argumentos construtores opcionais para a classe `Workflow`.

Nome do argumento	Tipo	Descrição
`Description`	`str`	Consulte Estrutura Workflow.
`DefaultRunProperties`	`dict`	Consulte Estrutura Workflow.
`OnSchedule`	`str`	Uma expressão `cron`.

Classe Methods

Todas as três classes incluem os seguintes métodos.

validate()

Valida as propriedades do objeto e, se forem encontrados erros, emite uma mensagem e sai. Não gera saída se não houver erros. Para a classe `Workflow`, chama a si mesma em cada entidade no fluxo de trabalho.

to_json()

Serializa o objeto em JSON. Também chama `validate()`. Para a classe `Workflow`, o objeto JSON inclui listas de trabalhos e crawlers e uma lista de acionadores gerados pelas especificações de dependência de trabalho e de crawler.

Esquemas de exemplo

Diversos projetos de esquema de exemplo estão disponíveis no repositório do Github de esquemas do AWS Glue. Tais amostras são apenas para referência e não se destinam à produção.

Os títulos dos projetos de exemplo são:

- Compaction (Compactação): esse blueprint cria um trabalho que compacta arquivos de entrada em blocos maiores com base no tamanho de arquivo desejado.
- Conversion (Conversão): esse blueprint converte arquivos de entrada em vários formatos de arquivo padrão no formato Apache Parquet, que é otimizado para workloads analíticas.
- Crawling Amazon S3 locations (Crawling de locais do Amazon S3): esse blueprint rastreia vários locais do Amazon S3 para adicionar tabelas de metadados ao Data Catalog.
- Custom connection to Data Catalog (Conexão personalizada com o Data Catalog): esse esquema acessa os armazenamentos de dados usando conectores personalizados do AWS Glue, lê os registros e preenche as definições no AWS Glue Data Catalog no esquema de registro.
- Encoding (Codificação): esse blueprint converte seus arquivos não UTF em arquivos codificados em UTF.
- Partitioning (Particionamento): esse blueprint cria um trabalho de particionamento que coloca os arquivos de saída em partições com base em chaves de partição específicas.
- Importing Amazon S3 data into a DynamoDB table (Importação de dados do Amazon S3 em uma tabela do DynamoDB): esse blueprint importa dados do Amazon S3 para uma tabela do DynamoDB.
- Standard table to governed (Tabela padrão a ser controlada): esse esquema importa uma tabela do AWSGlue Data Catalog para uma tabela do Lake Formation.

Registrar um esquema no AWS Glue

Após o desenvolvedor do AWS Glue codificar o esquema e carregar um arquivo ZIP no Amazon Simple Storage Service (Amazon S3), um administrador do AWS Glue deve registrar o esquema. Registrar o blueprint o torna disponível para uso.

Quando você registra um esquema, o AWS Glue copia o arquivo de esquema para um local reservado do Amazon S3. Em seguida, você pode excluir o arquivo do local de onde foi carregado.

Para registrar um blueprint, você precisa de permissões de leitura no local do Amazon S3 que contém o arquivo carregado. Você também precisa da permissão do AWS Identity and Access

Management (IAM) `glue:CreateBlueprint`. Para obter as permissões sugeridas para um administrador do AWS Glue que deve registrar, visualizar e manter esquemas, consulte Permissões de administrador do AWS Glue para esquemas.

Você pode registrar um esquema usando o console do AWS Glue, a API do AWS Glue ou a AWS Command Line Interface (AWS CLI).

Para registrar um blueprint (console)

1. Verifique se você tem permissões de leitura (`s3:GetObject`) no arquivo ZIP de blueprint no Amazon S3.
2. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.

   Faça login como um usuário que tem permissões para registrar um blueprint. Mude para a mesma região da AWS do bucket do Amazon S3 que contém o arquivo ZIP de esquema.
3. No painel de navegação, escolha Blueprints (Esquemas). Em seguida, na página Blueprints (Esquemas), escolha Add blueprint (Adicionar esquema).
4. Insira um nome de blueprint e uma descrição opcional.
5. Para o local do arquivo ZIP (S3), insira o caminho do Amazon S3 do arquivo ZIP de blueprint carregado. Inclua o nome do arquivo no caminho e o inicie com `s3://`.
6. (Opcional) adicione uma ou mais tags.
7. Escolha Add blueprint (Adicionar blueprint).

   A página Blueprints (Esquemas) retorna e mostra que o status do esquema é `CREATING`. Escolha o botão de atualização até que o status mude para `ACTIVE` ou `FAILED`.
8. Se o status for `FAILED`, selecione o blueprint e, no menu Actions (Ações), escolha View (Exibir).

   A página de detalhes mostra o motivo da falha. Se a mensagem do erro for “Unable to access object at location…” (Não foi possível acessar o objeto no local…) ou “Access denied on object at location…” (Acesso negado no objeto no local…), analise os seguintes requisitos:

   - O usuário ao qual você está conectado deve ter permissão de leitura no arquivo ZIP de blueprint no Amazon S3.
   - O bucket do Amazon S3 que contém o arquivo ZIP deve ter uma política de bucket que conceda permissão de leitura no objeto ao seu ID da conta da AWS. Para ter mais informações, consulte Desenvolver esquemas no AWS Glue.

- O bucket do Amazon S3 que você está usando deve estar na mesma região em que você se conectou no console.

9. Certifique-se de que os analistas de dados tenham permissões no blueprint.

   A política do IAM sugerida para analistas de dados é mostrada em Permissões de analista de dados para esquemas. Essa política concede `glue:GetBlueprint` em qualquer recurso. Se sua política for mais minuciosa no nível do recurso, conceda permissões aos analistas de dados nesse recurso recém-criado.

Para registrar um blueprint (AWS CLI)

1. Insira o comando a seguir.

```
aws glue create-blueprint --name <blueprint-name> [--description <description>] --
blueprint-location s3://<s3-path>/<archive-filename>
```

2. Para verificar o status do blueprint, insira o comando a seguir. Repita o comando até que o status vá para `ACTIVE` ou `FAILED`.

```
aws glue get-blueprint --name <blueprint-name>
```

   Se o status for `FAILED` e a mensagem de erro for “Unable to access object at location…” (Não foi possível acessar o objeto no local…) ou “Access denied on object at location…” (Acesso negado no objeto no local…), analise os seguintes requisitos:

   - O usuário ao qual você está conectado deve ter permissão de leitura no arquivo ZIP de blueprint no Amazon S3.
   - O bucket do Amazon S3 que contém o arquivo ZIP deve ter uma política de bucket que conceda permissão de leitura no objeto ao seu ID da conta da AWS. Para ter mais informações, consulte Publicar um esquema.
   - O bucket do Amazon S3 que você está usando deve estar na mesma região em que você se conectou no console.

Consulte também:

- Visão geral dos esquemas no AWS Glue

Visualizar esquemas no AWS Glue

Exiba um blueprint para revisar a descrição, o status e as especificações de parâmetros dele, e fazer o download do arquivo ZIP de esquema.

Você pode visualizar uma execução de esquema usando o console do AWS Glue, a API do AWS Glue ou a AWS Command Line Interface (AWS CLI).

Para exibir um blueprint (console)

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Blueprints (Esquemas).
3. Na página Blueprints (Esquemas), selecione um esquema. Em seguida, no menu Actions (Ações), escolha View (Exibir).

Para exibir um blueprint (AWS CLI)

- Insira o comando a seguir para exibir apenas o nome, a descrição e o status do blueprint. Substitua *<blueprint-name>* pelo nome do blueprint a ser exibido.

```
aws glue get-blueprint --name <blueprint-name>
```

A saída do comando é semelhante à seguinte.

```
{
    "Blueprint": {
        "Name": "myDemoBP",
        "CreatedOn": 1587414516.92,
        "LastModifiedOn": 1587428838.671,
        "BlueprintLocation": "s3://awsexamplebucket1/demo/
DemoBlueprintProject.zip",
        "Status": "ACTIVE"
    }
}
```

Insira o seguinte comando para visualizar também as especificações de parâmetros.

```
aws glue get-blueprint --name <blueprint-name>  --include-parameter-spec
```

A saída do comando é semelhante à seguinte.

```
{
    "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"
    }
}
```

Adicione o argumento `--include-blueprint` para incluir um URL na saída que você pode colar em seu navegador para baixar o arquivo ZIP de esquema que o AWS Glue armazenou.

Consulte também:

- Visão geral dos esquemas no AWS Glue

Atualizar um esquema no AWS Glue

Você pode atualizar um blueprint se tiver um script de layout revisado, um conjunto revisado de parâmetros do blueprint ou arquivos de suporte revisados. Atualizar um blueprint cria uma nova versão.

A atualização de um blueprint não afeta os fluxos de trabalho existentes criados dele.

Você pode atualizar um esquema usando o console do AWS Glue, a API do AWS Glue ou a AWS Command Line Interface (AWS CLI).

O procedimento a seguir pressupõe que o desenvolvedor do AWS Glue criou e carregou um arquivo ZIP de esquema atualizado no Amazon S3.

Para atualizar um blueprint (console)

1. Verifique se você tem permissões de leitura (`s3:GetObject`) no arquivo ZIP de blueprint no Amazon S3.
2. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.

   Faça login como um usuário que tem permissões para atualizar um blueprint. Mude para a mesma região da AWS do bucket do Amazon S3 que contém o arquivo ZIP de blueprint.
3. No painel de navegação, escolha Blueprints (Esquemas).
4. Na página Blueprints (Esquemas), selecione um esquema e, no menu Actions (Ações), escolha Edit (Editar).
5. Na página Edit a blueprint (Editar um blueprint), atualize Description (Descrição) ou ZIP archive location (S3) (Localização do arquivo ZIP [S3]) do blueprint. Certifique-se de incluir o nome do arquivo no caminho.
6. Escolha Salvar.

   A página Blueprints (Esquemas) retorna e mostra que o status do esquema é `UPDATING`. Escolha o botão de atualização até que o status mude para `ACTIVE` ou `FAILED`.
7. Se o status for `FAILED`, selecione o blueprint e, no menu Actions (Ações), escolha View (Exibir).

   A página de detalhes mostra o motivo da falha. Se a mensagem do erro for “Unable to access object at location…” (Não foi possível acessar o objeto no local…) ou “Access denied on object at location…” (Acesso negado no objeto no local…), analise os seguintes requisitos:

   - O usuário ao qual você está conectado deve ter permissão de leitura no arquivo ZIP de blueprint no Amazon S3.
   - O bucket do Amazon S3 que contém o arquivo ZIP deve ter uma política de bucket que conceda permissão de leitura no objeto ao seu ID da conta da AWS. Para ter mais informações, consulte Publicar um esquema.
   - O bucket do Amazon S3 que você está usando deve estar na mesma região em que você se conectou no console.

> **Note**
>
> Se a atualização falhar, a próxima execução do blueprint usará a versão mais recente dele que tenha sido registrada ou atualizada com êxito.

Para atualizar um blueprint (AWS CLI)

1. Insira o comando a seguir.

```
aws glue update-blueprint --name <blueprint-name> [--description <description>] --
blueprint-location s3://<s3-path>/<archive-filename>
```

2. Para verificar o status do blueprint, insira o comando a seguir. Repita o comando até que o status vá para `ACTIVE` ou `FAILED`.

```
aws glue get-blueprint --name <blueprint-name>
```

   Se o status for `FAILED` e a mensagem de erro for “Unable to access object at location…” (Não foi possível acessar o objeto no local…) ou “Access denied on object at location…” (Acesso negado no objeto no local…), analise os seguintes requisitos:

   - O usuário ao qual você está conectado deve ter permissão de leitura no arquivo ZIP de blueprint no Amazon S3.
   - O bucket do Amazon S3 que contém o arquivo ZIP deve ter uma política de bucket que conceda permissão de leitura no objeto ao seu ID da conta da AWS. Para ter mais informações, consulte Publicar um esquema.
   - O bucket do Amazon S3 que você está usando deve estar na mesma região em que você se conectou no console.

> **Consulte também**
>
> - Visão geral dos esquemas no AWS Glue

Criar um fluxo de trabalho com base em um esquema no AWS Glue

É possível criar um fluxo de trabalho do AWS Glue manualmente, adicionando um componente de cada vez, ou você pode criar um fluxo de trabalho a partir de um esquema do AWS Glue. O AWS Glue inclui esquemas para casos de uso comuns. Seus desenvolvedores do AWS Glue podem criar esquemas adicionais.

> **Important**
>
> Limite o número total de trabalhos, crawlers e gatilhos em um fluxo de trabalho para 100 ou menos. Se você incluir mais de 100, poderá receber erros ao tentar retomar ou interromper as execuções do fluxo de trabalho.

Ao usar um blueprint, você pode gerar rapidamente um fluxo de trabalho para um caso de uso específico com base no caso de uso generalizado definido pelo blueprint. Você define o caso de uso específico fornecendo valores para os parâmetros do blueprint. Por exemplo, um blueprint que particiona um conjunto de dados pode ter os caminhos de fonte e de destino do Amazon S3 como parâmetros.

O AWS Glue cria um fluxo de trabalho a partir de um esquema executando o esquema. A execução do blueprint salva os valores de parâmetro fornecidos e é usada para rastrear o progresso e o resultado da criação do fluxo de trabalho e de seus componentes. Ao solucionar problemas de um fluxo de trabalho, você pode exibir a execução do blueprint para determinar os valores de parâmetro do blueprint usados para criar um fluxo de trabalho.

Para criar e exibir fluxos de trabalho, você precisa de determinadas permissões do IAM. Para ver uma sugestão de política do IAM, consulte Permissões de analista de dados para esquemas.

Você pode criar um fluxo de trabalho a partir de um esquema usando o console do AWS Glue, a API do AWS Glue ou a AWS Command Line Interface (AWS CLI).

Para criar um fluxo de trabalho a partir de um blueprint (console)

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.

   Faça login como um usuário que tem permissões para criar um fluxo de trabalho.
2. No painel de navegação, escolha Blueprints (Esquemas).

3. Selecione um blueprint e, no menu Actions (Ações), escolha Create workflow (Criar fluxo de trabalho).
4. Na página Create a workflow from <blueprint-name> (Criar um fluxo de trabalho a partir de <nome-do-blueprint>), insira as seguintes informações:

   Parâmetros do blueprint

   Estes variam dependendo do design do blueprint. Para dúvidas sobre os parâmetros, consulte o desenvolvedor. Os esquemas geralmente contêm um parâmetro para o nome do fluxo de trabalho.

   IAM role (Perfil do IAM)

   A função que o AWS Glue assume para criar o fluxo de trabalho e seus componentes. A função deve ter permissões para criar e excluir fluxos de trabalho, trabalhos, crawlers e acionadores. Para obter uma política sugerida para a função, consulte Permissões para perfis de esquema.
5. Selecione Enviar.

   A página Blueprint Details (Detalhes do blueprint) é exibida, mostrando uma lista de execuções do blueprint na parte inferior.
6. Na lista de execuções do blueprint, verifique a execução do blueprint mais alta quanto ao status de criação do fluxo de trabalho.

   O status inicial é `RUNNING`. Escolha o botão de atualização até que o status mude para `SUCCEEDED` ou `FAILED`.
7. Execute um destes procedimentos:

   - Se o status de conclusão for `SUCCEEDED`, você pode ir para a página Workflows (Fluxos de trabalho), selecionar o fluxo de trabalho recém-criado e executá-lo. Antes de executar o fluxo de trabalho, você pode revisar o gráfico de design.
   - Se o status de conclusão for `FAILED`, selecione a execução do blueprint e, no menu Actions (Ações), escolha View (Exibir), para visualizar a mensagem de erro.

Para obter mais informações sobre fluxos de trabalho e esquemas, consulte os tópicos a seguir.

- Visão geral de fluxos de trabalho no AWS Glue
- Atualizar um esquema no AWS Glue

- Criar e desenvolver um fluxo de trabalho manualmente no AWS Glue

Visualizar execuções de esquemas no AWS Glue

Exiba uma execução de blueprint para ver as seguintes informações:

- Nome do fluxo de trabalho que foi criado.
- Valores de parâmetro do esquema que foram usados para criar o fluxo de trabalho.
- O status da operação de criação do fluxo de trabalho.

Você pode exibir uma execução de esquema usando o console do AWS Glue, a API do AWS Glue ou a AWS Command Line Interface (AWS CLI).

Para exibir uma execução de blueprint (console)

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Blueprints (Esquemas).
3. Na página Blueprints (Esquemas), selecione um esquema. Em seguida, no menu Actions (Ações), escolha View (Exibir).
4. Na parte inferior da página Blueprint Details (Detalhes do blueprint), selecione uma execução de blueprint e, no menu Actions (Ações), escolha View (Exibir).

Para exibir uma execução de blueprint (AWS CLI)

- Insira o comando a seguir. Substitua *<blueprint-name>* pelo nome do blueprint. Substitua *<blueprint-run-id>* pelo ID de execução do blueprint.

```
aws glue get-blueprint-run --blueprint-name <blueprint-name> --run-id <blueprint-run-id>
```

Consulte também:

- Visão geral dos esquemas no AWS Glue

AWS CloudFormation para AWS Glue

O AWS CloudFormation é um serviço que pode criar muitos recursos da AWS. O AWS Glue fornece operações de API para criar objetos no AWS Glue Data Catalog. No entanto, pode ser mais conveniente definir e criar objetos do AWS Glue e outros objetos de recurso relacionados da AWS em um arquivo de modelo do AWS CloudFormation. Em seguida, você pode automatizar o processo de criação de objetos.

O AWS CloudFormation fornece uma sintaxe simplificada, JSON (JavaScript Object Notation) ou YAML (YAML Ain't Markup Language), para expressar a criação de recursos da AWS. Você pode usar os modelos do AWS CloudFormation para definir os objetos do Data Catalog, como bancos de dados, tabelas, partições, crawlers, classificadores e conexões. Você também pode definir objetos ETL como trabalhos, acionadores e endpoints de desenvolvimento. Você cria um modelo que descreve todos os recursos da AWS desejados, e o AWS CloudFormation cuida do provisionamento e da configuração desses recursos para você.

Para obter mais informações, consulte O que é o AWS CloudFormation? e Trabalhar com modelos do AWS CloudFormation no Manual do usuário do AWS CloudFormation.

Se você planeja usar modelos do AWS CloudFormation compatíveis com o AWS Glue, como administrador, você deve conceder acesso ao AWS CloudFormation e às ações e aos serviços da AWS dos quais ele depende. Para conceder permissões para criar recursos do AWS CloudFormation, anexe a seguinte política aos usuários que trabalham com o AWS CloudFormation:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

A tabela a seguir contém as ações que um modelo do AWS CloudFormation pode realizar em seu nome. Ela inclui links para informações sobre os tipos de recursos da AWS e os tipos de propriedade que você pode adicionar a um modelo do AWS CloudFormation.

Recurso do AWS Glue	Modelo AWS CloudFormation	Exemplos do AWS Glue
Classificador	AWS::Glue::Classifier	Classificador grok, Classificador JSON, Classificador XML
Conexão	AWS::Glue::Connection	Conexão do MySQL
Crawler	AWS::Glue::Crawler	Crawler do Amazon S3, Crawler do MySQL
Banco de dados	AWS::Glue::Database	Banco de dados vazio, Banco de dados com tabelas
Endpoint de desenvolvimento	AWS::Glue::DevEndpoint	Endpoint de desenvolvimento
Trabalho	AWS::Glue::Job	Trabalho do Amazon S3, Trabalho do JDBC
Transformação de machine learning	AWS::Glue::MLTransform	Transformação de machine learning
Regras de qualidade de dados	AWS::Glue::DataQualityRuleset	Conjunto de regras de qualidade de dados, Conjunto de regras de qualidade de dados com o agendador do EventBridge
Partition	AWS::Glue::Partition	Partições de uma tabela
Tabela	AWS::Glue::Table	Tabela em um banco de dados
Trigger	AWS::Glue::Trigger	Acionador sob demanda, Acionador programado, Acionador condicional

Para começar, use os seguintes modelos de exemplo e personalize-os com seus próprios metadados. Em seguida, use o console do AWS CloudFormation para criar uma pilha do AWS CloudFormation para adicionar objetos ao AWS Glue e a todos os serviços associados. Muitos campos em um objeto do AWS Glue são opcionais. Esses modelos ilustram os campos que são obrigatórios ou necessários para um objeto do AWS Glue funcional.

Um modelo do AWS CloudFormation pode estar no formato JSON ou YAML. Nestes exemplos, o YAML é usado para facilitar a leitura. Os exemplos contêm comentários (#) para descrever os valores que são definidos nos modelos.

Os modelos do AWS CloudFormation podem incluir uma seção `Parameters`. Esta seção pode ser alterada no texto de exemplo ou quando o arquivo YAML for enviado para o console do AWS CloudFormation para criar uma pilha. A seção `Resources` do modelo contém a definição do AWS Glue e objetos relacionados. As definições de sintaxe de modelos do AWS CloudFormation pode conter propriedades que incluem uma sintaxe de propriedades mais detalhada. Nem todas as propriedades podem ser necessárias para criar um objeto do AWS Glue. Essas amostras mostram exemplos de valores para propriedades comuns para criar um objeto do AWS Glue.

Exemplo de modelo do AWS CloudFormation para um banco de dados do AWS Glue

Um banco de dados do AWS Glue no Data Catalog contém tabelas de metadados. O banco de dados consiste em poucas propriedades e pode ser criado no Data Catalog com um modelo do AWS CloudFormation. O modelo de exemplo a seguir é fornecido para você começar e para ilustrar o uso de pilhas do AWS CloudFormation com o AWS Glue. O único recurso criado pelo modelo de exemplo é um banco de dados chamado `cfn-mysampledatabase`. Você pode alterá-lo ao editar o texto do exemplo ou ao alterar o valor no console do AWS CloudFormation quando você enviar o YAML.

Veja a seguir exemplos de valores para as propriedades comuns para criar um banco de dados do AWS Glue. Para obter mais informações sobre o modelo de banco de dados do AWS CloudFormation para o 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
```

Exemplo de modelo do AWS CloudFormation para um banco de dados, uma tabela e uma partição do AWS Glue

Uma tabela do AWS Glue contém os metadados que definem a estrutura e o local dos dados que você deseja processar com seus scripts de ETL. Em uma tabela, você pode definir partições para paralelizar o processamento de seus dados. Uma partição é um bloco de dados definido com uma chave. Por exemplo, usando o mês como uma chave, todos os dados de janeiro estão contidos na mesma partição. No AWS Glue, bancos de dados podem conter tabelas, e tabelas podem conter partições.

O exemplo a seguir mostra como preencher um banco de dados, uma tabela e partições usando um modelo do AWS CloudFormation. O formato de dados base é `csv` e delimitado por uma vírgula (,). Como um banco de dados deve existir antes de poder conter uma tabela, e uma tabela deve existir antes que as partições possam ser criadas, o modelo usa a instrução `DependsOn` para definir a dependência desses objetos quando eles são criados.

Os valores neste exemplo definem uma tabela que contém dados de voo de um bucket do Amazon S3 publicamente disponível. Para ilustração, apenas algumas colunas dos dados e uma chave de particionamento são definidas. Quatro partições também são definidas no Data Catalog. Alguns campos para descrever o armazenamento do banco de dados também são mostrados nos campos `StorageDescriptor`.

```
---
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
```

Exemplo de modelo do AWS CloudFormation para um classificador grok do AWS Glue

Um classificador do AWS Glue determina o esquema dos seus dados. Um tipo de classificador personalizado usa um padrão grok para corresponder aos seus dados. Se o padrão corresponder, o classificador personalizado será usado para criar o esquema da tabela e definir `classification` como o valor definido na definição do classificador.

Este exemplo cria um classificador que cria um esquema com uma coluna denominada `message` e define a classificação para `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
```

Exemplo de modelo do AWS CloudFormation para um classificador JSON do AWS Glue

Um classificador do AWS Glue determina o esquema dos seus dados. Um tipo de classificador personalizado usa uma string `JsonPath` que define os dados JSON para o classificador classificar. O AWS Glue oferece suporte a um subconjunto de operadores para `JsonPath`, conforme descrito em Gravando classificadores personalizados JsonPath.

Se o padrão for correspondente, o classificador personalizado será usado para criar o esquema da tabela.

Este exemplo cria um classificador que cria um esquema com cada registro na matriz `Records3` em um 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[*]
```

Exemplo de modelo do AWS CloudFormation para um classificador XML do AWS Glue

Um classificador do AWS Glue determina o esquema dos seus dados. Um tipo de classificador personalizado especifica uma tag XML para designar o elemento que contém cada registro em um documento XML que está sendo analisado. Se o padrão corresponder, o classificador personalizado

será usado para criar o esquema da tabela e definir `classification` como o valor definido na definição do classificador.

Este exemplo cria um classificador que cria um esquema com uma registro na tag `Record` e define a classificação como `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>
```

Exemplo de modelo do AWS CloudFormation de um crawler do AWS Glue para o Amazon S3

Um crawler do AWS Glue cria tabelas de metadados no Data Catalog que correspondem aos seus dados. Você pode usar essas definições de tabela como origens e destinos em trabalhos de ETL.

Este exemplo cria um crawler, a função do IAM necessária e um banco de dados do AWS Glue no Data Catalog. Quando esse crawler é executado, ele assume a função do IAM e cria uma tabela no banco de dados para os dados de voos públicos. A tabela é criada com o prefixo "`cfn_sample_1_`". A função do IAM criada por esse modelo permite permissões globais, caso você queira criar uma função personalizada. Nenhum classificador personalizado é definido por esse classificador. Os classificadores integrados do AWS Glue são usados por padrão.

Quando você envia esse exemplo para o console do AWS CloudFormation, é necessário confirmar se deseja criar a função do 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\"}}}"
```

Exemplo de modelo do AWS CloudFormation para uma conexão do AWS Glue

Uma conexão do AWS Glue no Data Catalog contém as informações do JDBC e de rede que são necessárias para se conectar a um banco de dados JDBC. Essas informações são usadas quando você se conecta a um banco de dados JDBC para rastrear ou executar trabalhos de ETL.

Este exemplo cria uma conexão com um banco de dados MySQL do Amazon RDS chamado `devdb`. Quando essa conexão é usada, uma função do IAM, as credenciais do banco de dados e os valores de conexão de rede também devem ser fornecidos. Consulte as informações dos campos necessários no modelo.

```
---
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
```

Exemplo de modelo do AWS CloudFormation para um crawler do AWS Glue para o JDBC

Um crawler do AWS Glue cria tabelas de metadados no Data Catalog que correspondem aos seus dados. Você pode usar essas definições de tabela como origens e destinos em trabalhos de ETL.

Este exemplo cria um crawler, a função do IAM necessária e um banco de dados do AWS Glue no Data Catalog. Quando esse crawler é executado, ele assume a função do IAM e cria uma tabela no banco de dados para os dados de voo públicos que foram armazenados em um banco de dados MySQL. A tabela é criada com o prefixo "`cfn_jdbc_1_`". A função do IAM criada por esse modelo permite permissões globais, caso você queira criar uma função personalizada. Nenhum classificador personalizado pode ser definido para dados JDBC. Os classificadores integrados do AWS Glue são usados por padrão.

Quando você envia esse exemplo para o console do AWS CloudFormation, é necessário confirmar se deseja criar a função do 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\"}}}"
```

Exemplo do modelo do AWS CloudFormation de um trabalho do AWS Glue para o Amazon S3 ao Amazon S3

Um trabalho do AWS Glue no Data Catalog contém os valores de parâmetros que são necessários para executar um script no AWS Glue.

Este exemplo cria um trabalho que lê dados de voo de um bucket do Amazon S3 no formato `csv` e os grava em um arquivo Parquet do Amazon S3. O script que é executado por esse trabalho já deve existir. Você pode gerar um script de ETL para seu ambiente com o console do AWS Glue. Quando esse trabalho é executado, uma função do IAM com as permissões corretas também deve ser fornecida.

Os valores de parâmetro comuns são exibidos no modelo. Por exemplo, o padrão é 5 para `AllocatedCapacity` (DPUs).

```
---
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
```

Exemplo de modelo do AWS CloudFormation de um trabalho do AWS Glue para JDBC para o Amazon S3

Um trabalho do AWS Glue no Data Catalog contém os valores de parâmetros que são necessários para executar um script no AWS Glue.

Este exemplo cria um trabalho que lê dados de voo de um banco de dados MySQL do JDBC, conforme definido pela conexão denominada `cfn-connection-mysql-flights-1` e os grava em

um arquivo Parquet do Amazon S3. O script que é executado por esse trabalho já deve existir. Você pode gerar um script de ETL para seu ambiente com o console do AWS Glue. Quando esse trabalho é executado, uma função do IAM com as permissões corretas também deve ser fornecida.

Os valores de parâmetro comuns são exibidos no modelo. Por exemplo, o padrão é 5 para `AllocatedCapacity` (DPUs).

```
---
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
```

Exemplo de modelo do AWS CloudFormation para um acionador sob demanda do AWS Glue

Um acionador do AWS Glue no Data Catalog contém os valores de parâmetros que são necessários para iniciar uma execução de trabalho quando esse acionador é disparado. Um acionador sob demanda é acionado quando você o ativa.

Este exemplo cria um acionador sob demanda que inicia um trabalho denominado `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:
```

Exemplo de modelo do AWS CloudFormation para um acionador programado do AWS Glue

Um acionador do AWS Glue no Data Catalog contém os valores de parâmetros que são necessários para iniciar uma execução de trabalho quando esse acionador é disparado. Um acionador programado é acionado quando está habilitado e o temporizador cron é exibido.

Este exemplo cria um acionador programado que inicia um trabalho denominado `cfn-job-S3-to-S3-1`. O temporizador é uma expressão cron para executar o trabalho a cada 10 minutos em dias da 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:
```

Exemplo de modelo do AWS CloudFormation para um acionador condicional do AWS Glue

Um acionador do AWS Glue no Data Catalog contém os valores de parâmetros que são necessários para iniciar uma execução de trabalho quando esse acionador é disparado. Um acionador condicional é acionado quando é ativado e suas condições são atendidas, como um trabalho sendo concluído com êxito.

Este exemplo cria um acionador condicional que inicia um trabalho denominado `cfn-job-S3-to-S3-1`. Esse trabalho começa quando o trabalho denominado `cfn-job-S3-to-S3-2` é concluído com êxito.

```
---
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
```

Exemplo de modelo do AWS CloudFormation para um endpoint de desenvolvimento do AWS Glue

Uma transformação de machine learning do AWS Glue é uma transformação personalizada para limpar seus dados. No momento, existe uma transformação disponível chamada FindMatches. A transformação FindMatches permite identificar registros duplicados ou correspondentes no seu conjunto de dados, mesmo quando os registros não têm um identificador exclusivo comum e quando não há campos com uma correspondência exata.

Essa amostra cria uma transformação de machine learning. Para obter mais informações sobre os parâmetros necessários para criar uma transformação de machine learning, consulte Correspondência de registros com o FindMatches do 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
```

Exemplo de modelo do AWS CloudFormation para um conjunto de regras do AWS Glue Data Quality

Um conjunto de regras de qualidade de dados do AWS Glue contém regras que podem ser avaliadas em uma tabela no catálogo de dados. Depois que o conjunto de regras é colocado na tabela de destino, você pode acessar o catálogo de dados e executar uma avaliação que executa seus dados de acordo com as regras do conjunto de regras. Essas regras podem variar desde a avaliação do número de linhas até a avaliação da integridade referencial em seus dados.

O exemplo a seguir é um modelo do CloudFormation que cria um conjunto de regras com uma variedade de regras na tabela 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
```

Modelo da AWS CloudFormation de amostra para um conjunto de regras do AWS Glue Data Quality com o agendador do EventBridge

Um conjunto de regras de qualidade de dados do AWS Glue contém regras que podem ser avaliadas em uma tabela no catálogo de dados. Depois que o conjunto de regras é colocado na tabela de destino, você pode acessar o catálogo de dados e executar uma avaliação que executa seus dados de acordo com as regras do conjunto de regras. Em vez de precisar acessar manualmente o catálogo de dados para avaliar o conjunto de regras, você também pode adicionar um agendador do EventBridge em nosso modelo do CloudFormation para agendar essas avaliações do conjunto de regras para você em um intervalo cronometrado.

O exemplo a seguir é um modelo do CloudFormation que cria um conjunto de regras de qualidade de dados e um agendador do EventBridge para avaliar o conjunto de regras mencionado acima a 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" ] }
        '
```

Exemplo de modelo do AWS CloudFormation para um endpoint de desenvolvimento do AWS Glue

Um endpoint de desenvolvimento do AWS Glue é um ambiente que você pode usar para desenvolver e testar seus scripts do AWS Glue.

Este exemplo cria um endpoint de desenvolvimento com os valores mínimos de parâmetros de rede necessários para criá-lo com êxito. Para obter mais informações sobre os parâmetros necessários para configurar um endpoint de desenvolvimento, consulte Configurar redes para desenvolvimento para o AWS Glue.

Você fornece um ARN (nome do recurso da Amazon) da função do IAM existente para criar o endpoint de desenvolvimento. Forneça uma chave pública RSA válida e mantenha a chave privada correspondente disponível se você planeja criar um servidor de notebook no endpoint de desenvolvimento.

> **Note**
>
> Todo servidor de notebook que criar associado a um endpoint de desenvolvimento será gerenciado por você. Portanto, se você excluir o endpoint de desenvolvimento, deverá excluir a pilha do AWS CloudFormation no console do AWS CloudFormation se quiser excluir o servidor de notebook.

```
---
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
```

Guia de programação do AWS Glue

Um script contém o código que extrai os dados das fontes, os transforma e os carrega nos destinos. O AWS Glue executa um script quando inicia um trabalho.

Os scripts de ETL do AWS Glue são codificados em Python ou Scala. Embora todos os tipos de trabalho possam ser escritos em Python, os trabalhos do AWS Glue for Spark também podem ser escritos também no Scala. Quando você gera automaticamente a lógica do código fonte para um trabalho no AWS Glue Studio, o script é criado. Você pode editar este script ou fornecer seu próprio script para processar seu trabalho de ETL.

Fornecer seus próprios scripts personalizados

Os scripts executam o trabalho de extração, transformação e carregamento (ETL) no AWS Glue. Um script é criado quando você gera automaticamente a lógica do código fonte para um trabalho. É possível editar este script gerado ou fornecer seu próprio script personalizado.

Para fornecer seu próprio script personalizado no AWS Glue, siga estas etapas gerais:

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Escolha a guia Trabalhos de ETL e, em seguida, visualize a seção Criar trabalho. Escolha uma opção de editor de scripts.
3. Em This job runs, escolha uma das seguintes opções:
   - Criar um novo script com código padronizado
   - Carregar e editar um script existente
4. Na tela Propriedades do trabalho, escolha o perfil do IAM exigido para que seu script personalizado seja executado. Para ter mais informações, consulte Gerenciamento de identidade e acesso para AWS Glue.
5. Escolha as conexões que seu script referencia. Esses objetos são necessários para se conectar aos armazenamentos de dados JDBC necessários.

   Uma interface de rede elástica é uma interface de rede virtual que você pode vincular a uma instância em uma nuvem virtual privada (VPC). Escolha a interface de rede elástica necessária para conexão com o armazenamento de dados que é usado no script.

6. Forneça as configurações adicionais, incluindo parâmetros, específicos para seu tipo de trabalho. Para obter mais informações sobre a configuração do tipo de trabalho, consulte a seção Criar trabalhos ETL visuais com o AWS Glue Studio.
7. Na guia Script, cole ou escreva o script personalizado.

Use o conteúdo dessa seção para orientar o processo de escrever o script personalizado.

Para obter mais informações sobre como adicionar trabalhos no AWS Glue, consulte Criar trabalhos ETL visuais com o AWS Glue Studio.

Para um guia detalhado, consulte o tutorial Add job no console do AWS Glue.

Programar scripts do Spark

O AWS Glue facilita a gravação ou a geração automática de scripts de extração, transformação e carregamento (ETL), além de testá-los e executá-los. Esta seção descreve as extensões ao Apache Spark que o AWS Glue apresentou e fornece exemplos de como codificar e executar scripts de ETL em Python e Scala.

> **Important**
>
> Versões diferentes do AWS Glue oferecem suporte a diferentes versões do Apache Spark. Seu script personalizado deve ser compatível com a versão compatível do Apache Spark. Para obter mais informações sobre as versões do AWS Glue, consulte Glue version job property.

Tópicos

- Tutorial: Escrevendo um script do AWS Glue for Spark
- Programe scripts AWS Glue ETL em PySpark
- Programando scripts de ETL do AWS Glue no Scala
- Recursos e otimizações para programação de scripts de ETL do AWS Glue para Spark

Tutorial: Escrevendo um script do AWS Glue for Spark

Este tutorial apresenta o processo de escrever scripts do AWS Glue. É possível executar scripts com base em um cronograma com trabalhos ou interativamente, com sessões interativas. Para

mais informações sobre trabalhos, consulte Criar trabalhos ETL visuais com o AWS Glue Studio. Para obter mais informações sobre sessões interativas, consulte the section called “Visão geral das sessões interativas do AWS Glue”.

O editor visual AWS Glue Studio oferece uma interface gráfica sem código para criar trabalhos do AWS Glue. AWS Glue scripts em trabalhos visuais. Eles dão acesso ao conjunto expandido de ferramentas disponíveis para trabalhar com os programas do Apache Spark. Você pode acessar as APIs nativas do Spark, bem como as bibliotecas do AWS Glue que facilitam fluxos de trabalho de extração, transformação e carregamento (ETL) de dentro de um script do Glue. AWS

Neste tutorial, você extrai, transforma e carrega um conjunto de dados de multas de estacionamento. O script que faz esse trabalho é idêntico em forma e função ao gerado em Making ETL easy with AWS Glue Studio no AWS Big Data Blog, que apresenta o editor visual AWS Glue Studio. Ao executar esse script em um trabalho, você pode compará-lo com trabalhos visuais e ver como os scripts AWS Glue ETL funcionam. Assim, você fica preparado para usar funcionalidades adicionais que ainda não estão disponíveis em trabalhos visuais.

Neste tutorial você usa a linguagem e as bibliotecas do Python. Há funcionalidades similares disponíveis no Scala. Depois de ler este tutorial, você poderá gerar e inspecionar um exemplo de script Scala para entender como realizar o processo de criação do script ETL do Scala AWS Glue.

Pré-requisitos

Este tutorial tem os seguintes pré-requisitos:

- Os mesmos pré-requisitos da postagem do blog do AWS Glue Studio, que instrui você a executar um modelo. AWS CloudFormation

  Este modelo usa o AWS Glue Data Catalog para gerenciar o conjunto de dados de multas de estacionamento disponível em `s3://aws-bigdata-blog/artifacts/gluestudio/`. Ele cria os seguintes recursos que serão referenciados:
- AWS Glue StudioRole(Função): o perfil do IAM para a execução de trabalhos da AWS Glue
- AWS Glue StudioAmazon S3Bucket: nome do bucket do Amazon S3 para armazenamento de arquivos relacionados ao blog
- AWS Glue StudioTicketsYYZDB: banco de dados do AWS Glue Data Catalog
- AWS Glue StudioTableTickets— Tabela do catálogo de dados para usar como fonte
- AWS Glue StudioTableTrials— Tabela do catálogo de dados para usar como fonte

- AWS Glue StudioParkingTicketCount — Tabela do catálogo de dados para usar como destino
- O script gerado na postagem do blog do AWS Glue Studio. Caso a publicação no blog mude, o script também está disponível no texto a seguir.

Gerar um exemplo de script

Você pode usar o editor visual AWS Glue Studio como uma poderosa ferramenta de geração de código para criar uma estrutura para o script que você deseja escrever. Você usará essa ferramenta para criar um exemplo de script.

Se você quiser pular essas etapas, o script será fornecido.

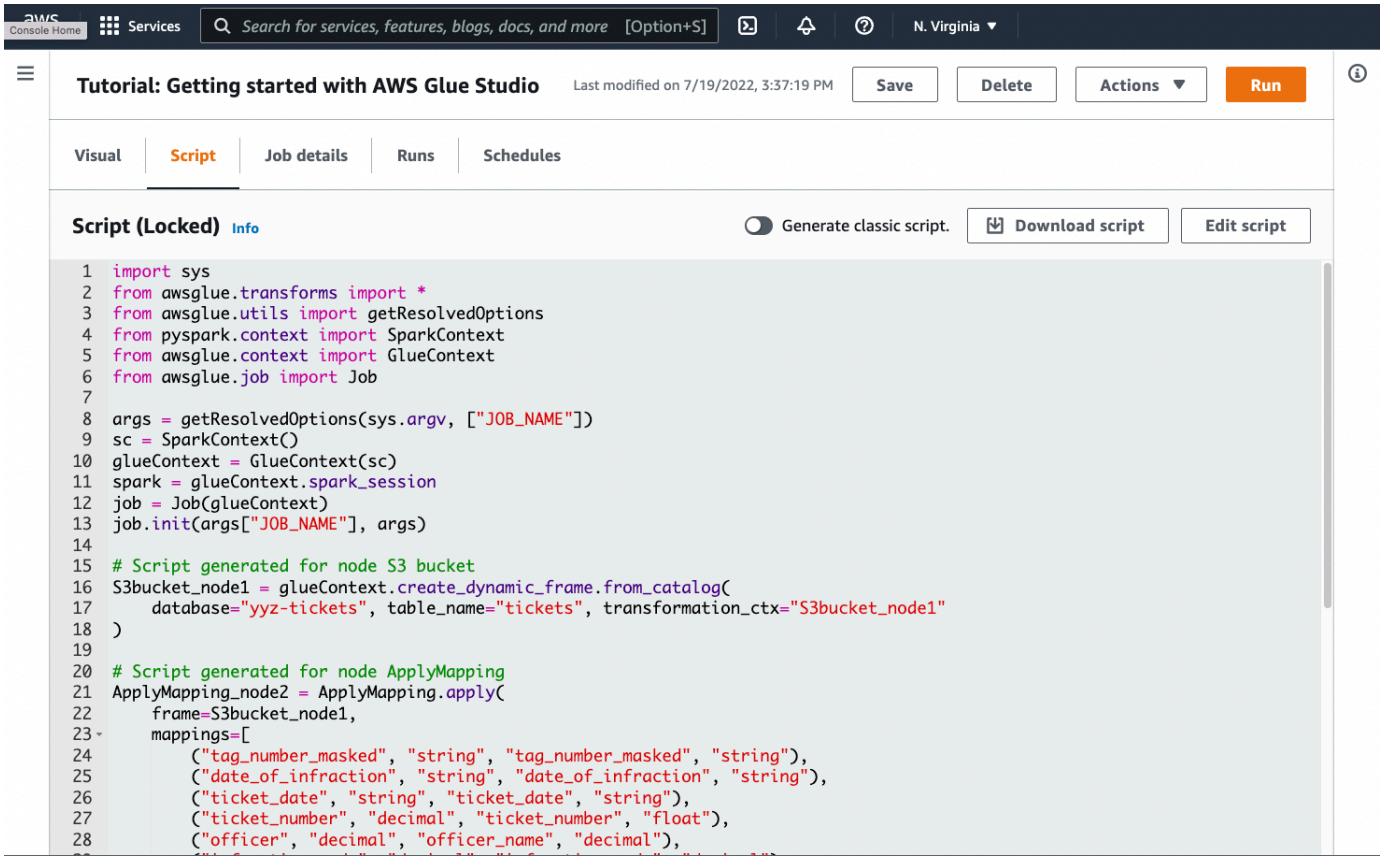
Exemplo de script do tutorial

```
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 gerar um exemplo de script

1. Conclua o tutorial do AWS Glue Studio. Para concluir este tutorial, consulte Criação de um trabalho no AWS Glue Studio a partir de um exemplo de trabalho.
2. Navegue até a guia Script na página do trabalho, conforme mostrado na captura de tela a seguir:



3. Copie o conteúdo completo da guia Script. Ao definir a linguagem do script em Job details (Detalhes do trabalho), você pode alternar entre gerar código para Python ou Scala.

Etapa 1. Criar um trabalho e colar seu script

Nesta etapa, você cria uma tarefa AWS Glue no AWS Management Console. Isso configura uma configuração que permite que o AWS Glue execute seu script. Essa ação também cria um local para você armazenar e editar seu script.

Como criar um trabalho

1. No AWS Management Console, navegue até a página inicial do AWS Glue.
2. No painel de navegação lateral, escolha Jobs (Trabalhos).
3. Escolha Spark script editor (Editor de scripts do Spark) em Create job (Criar trabalho) e, em seguida, escolha Create (Criar).
4. Opcional: cole o texto completo do seu script no painel Script (Script). Você também tem a opção de acompanhar o tutorial.

Etapa 2. Importe bibliotecas do AWS Glue

Você precisa configurar seu script para interagir com código e configuração que sejam definidos fora do script. Esse trabalho é feito nos bastidores no AWS Glue Studio.

Nesta etapa, você executará as ações a seguir.

- Importe e inicialize um objeto `GlueContext`. Do ponto de vista de criação do script, essa é a importação mais importante. Essa ação expõe métodos padrão para definir conjuntos de dados de origem e destino, que são o ponto de partida para qualquer script de ETL. Para saber mais a respeito da classe `GlueContext`, consulte GlueContext classe.
- Inicialize um `SparkContext` e `SparkSession`. Isso permite que você configure o mecanismo Spark disponível na tarefa AWS Glue. Você não precisará usá-los diretamente nos scripts introdutórios do AWS Glue.
- Chame `getResolvedOptions` para preparar seus argumentos de trabalho para uso no script. Para obter mais informações sobre como resolver parâmetros de trabalho, consulte the section called “getResolvedOptions”.
- Inicialize um `Job`. O `Job` objeto define a configuração e rastreia o estado de vários recursos opcionais do AWS Glue. Seu script pode ser executado sem um objeto de `Job`, mas a prática recomendada é inicializá-lo para que você não se confunda se esses recursos forem integrados posteriormente.

  Um desses recursos está nos marcadores de trabalho, que você pode configurar opcionalmente neste tutorial. Saiba mais sobre marcadores de trabalho na seção a seguir, the section called “Opcional: habilitar marcadores de trabalho”.

Neste procedimento, você vai escrever o código a seguir. Esse código é uma parte do script de exemplo gerado.

```
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 AWS Glue

- Copie essa seção de código e cole-a no editor de Script.

> **Note**
>
> Talvez você considere a ação de copiar o código uma prática inadequada de engenharia. Neste tutorial, sugerimos isso para incentivá-lo a nomear consistentemente suas principais variáveis em todos os scripts de ETL do AWS Glue.

Etapa 3. Extrair dados de uma fonte

Em qualquer processo de ETL, primeiro é necessário definir um conjunto de dados de origem que você deseje alterar. No editor visual do AWS Glue Studio, você fornece essas informações criando um nó de origem.

Nesta etapa, você fornece ao método `create_dynamic_frame.from_catalog` um `database` e um `table_name` para extrair dados de uma origem configurada no AWS Glue Data Catalog.

Na etapa anterior, você inicializou um objeto `GlueContext`. Você usa esse objeto para encontrar métodos que são usados para configurar origens, como `create_dynamic_frame.from_catalog`.

Neste procedimento, você escreverá o código a seguir usando `create_dynamic_frame.from_catalog`. Esse código é uma parte do script de exemplo gerado.

```
S3bucket_node1 = glueContext.create_dynamic_frame.from_catalog(
    database="yyz-tickets", table_name="tickets", transformation_ctx="S3bucket_node1"
    )
```

Para extrair dados de uma fonte

1. Examine a documentação para encontrar um método `GlueContext` para extrair dados de uma fonte definida no AWS Glue Data Catalog. Esses métodos estão documentados em the section called “GlueContext”. Escolha o método create_dynamic_frame.from_catalog. Chame esse método em `glueContext`.

2. Confira a documentação para `create_dynamic_frame.from_catalog`. Esse método requer os parâmetros `database` e `table_name`. Forneça os parâmetros necessários para `create_dynamic_frame.from_catalog`.

   O AWS Glue Data Catalog armazena informações sobre a localização e o formato dos seus dados de origem e foi configurado na seção de pré-requisitos. Você não precisa fornecer diretamente essas informações ao seu script.

3. Opcional: forneça o parâmetro `transformation_ctx` ao método para oferecer suporte aos marcadores do trabalho. Você pode aprender sobre marcadores de trabalho na seção a seguir, the section called “Opcional: habilitar marcadores de trabalho”.

> **Note**
>
> Métodos comuns para extrair dados
> the section called “create_dynamic_frame_from_catalog”é usado para se conectar às tabelas no AWS Glue Data Catalog.
> Caso precise fornecer diretamente ao seu trabalho uma configuração que descreva a estrutura e o local da sua fonte, consulte o método the section called “create_dynamic_frame_from_options”. Você precisará fornecer parâmetros mais detalhados descrevendo seus dados do que seria necessário ao usar `create_dynamic_frame.from_catalog`.
> Consulte a documentação complementar sobre `format_options` e `connection_parameters` para identificar os parâmetros necessários. Para obter uma explicação sobre como fornecer informações sobre o formato de dados de origem ao script, consulte the section called “Opções de formato de dados”. Para obter uma explicação sobre como fornecer ao script informações sobre a localização dos dados de origem, consulte the section called “Parâmetros de conexão”.
> Se você estiver lendo informações de uma fonte de transmissão, forneça ao seu trabalho as informações da fonte por meio do método the section called “create_data_frame_from_catalog” ou the section called “create_data_frame_from_options”. Observe que esses métodos retornam `DataFrames` do Apache Spark.
> Nosso código gerado chama `create_dynamic_frame.from_catalog`, enquanto a documentação de referência faz referência a `create_dynamic_frame_from_catalog`. Em última análise, esses métodos chamam o mesmo código e estão incluídos para que você possa escrever um código mais limpo. Você pode verificar isso visualizando o código-fonte do nosso wrapper Python, disponível em aws-glue-libs.

Etapa 4. Transformar dados com o AWS Glue

Depois de extrair os dados da fonte em um processo de ETL, você precisa descrever como deseja alterar seus dados. Você fornece essas informações criando um nó Transform no editor visual do AWS Glue Studio.

Nesta etapa, forneça ao método `ApplyMapping` um mapa dos nomes e tipos de campos atuais e desejados para transformar seu `DynamicFrame`.

Você executa as seguintes transformações.

- Descartar as quatro chaves `location` e `province`.
- Alterar o nome de `officer` para `officer_name`.
- Alterar o tipo de `ticket_number` e `set_fine_amount` para `float`.

`create_dynamic_frame.from_catalog` fornece a você um objeto `DynamicFrame`. `DynamicFrame`A representa um conjunto de dados no AWS Glue. AWS As transformações Glue são operações que mudam`DynamicFrames`.

> **Note**
>
> O que é um `DynamicFrame`?
> Um `DynamicFrame` é uma abstração que permite conectar um conjunto de dados com uma descrição dos nomes e tipos de entradas nos dados. No Apache Spark, existe uma abstração semelhante chamada de. DataFrame Para obter uma explicação DataFrames, consulte o Guia SQL do Spark.
> Com `DynamicFrames`, você pode descrever esquemas de conjuntos de dados dinamicamente. Considere um conjunto de dados com uma coluna de preços, em que algumas entradas armazenam o preço como uma sequência de caracteres e outras armazenam o preço como um dobro. AWS O Glue calcula um esquema on-the-fly — ele cria um registro autodescritivo para cada linha.
> Campos inconsistentes (como preço) são representados explicitamente com um tipo (`ChoiceType`) no esquema do quadro. Você pode lidar com campos inconsistentes descartando-os com `DropFields` ou solucionando-os com `ResolveChoice`. Essas são transformações que estão disponíveis no `DynamicFrame`. Em seguida, você pode gravar seus dados de volta em seu data lake com `writeDynamicFrame`.

Você pode chamar muitas das mesmas transformações a partir de métodos na classe `DynamicFrame`, que pode levar a scripts mais fáceis de ler. Para obter mais informações sobre o `DynamicFrame`, consulte the section called “DynamicFrame”.

Neste procedimento, você escreverá o código a seguir usando `ApplyMapping`. Esse código é uma parte do script de exemplo gerado.

```
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 dados com o AWS Glue

1. Examine a documentação para identificar uma transformação para alterar e descartar campos. Para obter detalhes, consulte the section called “GlueTransform”. Escolha a transformação `ApplyMapping`. Para obter mais informações sobre o `ApplyMapping`, consulte the section called “ApplyMapping”. Chame `apply` no objeto de transformação `ApplyMapping`.

   > **Note**
   >
   > O que é o `ApplyMapping`?
   > `ApplyMapping` pega um `DynamicFrame` e o transforma. Ele obtém uma lista de tuplas que representa transformações nos campos, um “mapeamento”. Os dois primeiros elementos de tupla, um nome e um tipo de campo, são usados para identificar um campo no quadro. Os dois segundos parâmetros também são um nome e um tipo de campo.

ApplyMapping converte o campo de origem no nome de destino e digite um novo`DynamicFrame`, que ele retorna. Campos que não forem fornecidos serão descartados no valor retornado.
Em vez de chamar `apply`, você pode chamar a mesma transformação com o método `apply_mapping` no `DynamicFrame` para criar um código mais fluente e legível. Para ter mais informações, consulte the section called “apply_mapping”.

2. Para identificar os parâmetros necessários, confira a documentação sobre `ApplyMapping`. Consulte the section called “ApplyMapping”. Você descobrirá que esse método requer os parâmetros `frame` e `mappings`. Forneça os parâmetros necessários para `ApplyMapping`.
3. Opcional: forneça `transformation_ctx` ao método para dar suporte a marcadores de trabalho. Você pode aprender sobre marcadores de trabalho na seção a seguir, the section called “Opcional: habilitar marcadores de trabalho”.

> **Note**
>
> Funcionalidade do Apache Spark
> Fornecemos as transformações para agilizar os fluxos de trabalho de ETL em seu trabalho. Você também tem acesso às bibliotecas que estão disponíveis em um programa do Spark no seu trabalho, criadas para uso mais geral. Para usá-las, converta entre `DynamicFrame` e `DataFrame`.
> É possível criar um `DataFrame` com the section called “toDF”. Em seguida, você pode usar os métodos disponíveis no DataFrame para transformar seu conjunto de dados. Para obter mais informações sobre esses métodos, consulte DataFrame. Em seguida, você pode converter para trás usando as operações AWS Glue para carregar sua moldura em um alvo. the section called “fromDF”

Etapa 5. Carregar dados em um destino

Depois de transformar seus dados, normalmente você armazena os dados transformados em um local diferente da fonte. Você executa essa operação criando um nó de destino no editor visual do AWS Glue Studio.

Nesta etapa, você fornecerá ao método `write_dynamic_frame.from_options` um `connection_type`, `connection_options`, `format` e `format_options` para carregar dados em um bucket de destino no Amazon S3.

Na Etapa 1, você inicializou um objeto `GlueContext`. No AWS Glue, é aqui que você encontrará métodos usados para configurar destinos, assim como as fontes.

Neste procedimento, você escreverá o código a seguir usando `write_dynamic_frame.from_options`. Esse código é uma parte do script de exemplo gerado.

```
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 carregar dados em um destino

1. Confira a documentação a fim de encontrar um método para carregar dados em um bucket de destino do Amazon S3. Esses métodos estão documentados em the section called “GlueContext”. Escolha o método the section called “write_dynamic_frame_from_options”. Chame esse método em `glueContext`.

   > **Note**
   >
   > Métodos comuns para carregar dados
   > `write_dynamic_frame.from_options` é o método mais comum usado para carregar dados. Ele suporta todos os alvos que estão disponíveis no AWS Glue.
   > Se você estiver gravando em um destino JDBC definido em uma conexão AWS Glue, use o the section called “write_dynamic_frame_from_jdbc_conf” método. AWS As conexões Glue armazenam informações sobre como se conectar a uma fonte de dados. Isso elimina a necessidade de fornecer essas informações no `connection_options`. No entanto, você ainda precisa usar `connection_options` para fornecer `dbtable`.
   > `write_dynamic_frame.from_catalog` não é um método comum para carregar dados. Esse método atualiza o AWS Glue Data Catalog sem atualizar o conjunto de dados subjacente e é usado em combinação com outros processos que alteram o conjunto de dados subjacente. Para ter mais informações, consulte the section called “Atualizar o esquema e adicionar novas partições”.

2. Confira a documentação para the section called “write_dynamic_frame_from_options”. Esse método requer `frame`, `connection_type`, `format`, `connection_options` e `format_options`. Chame esse método em `glueContext`.

   a. Consulte a documentação suplementar sobre `format_options` e `format` para identificar os parâmetros necessários. Para obter uma explicação sobre os formatos de dados, consulte the section called “Opções de formato de dados”.

   b. Consulte a documentação suplementar sobre `connection_type` e `connection_options` para identificar os parâmetros necessários. Para obter uma explicação sobre as conexões, consulte the section called “Parâmetros de conexão”.

   c. Forneça os parâmetros necessários para `write_dynamic_frame.from_options`. Esse método tem uma configuração similar a `create_dynamic_frame.from_options`.

3. Opcional: forneça `transformation_ctx` a `write_dynamic_frame.from_options` para dar suporte a marcadores de trabalho. Você pode aprender sobre marcadores de trabalho na seção a seguir, the section called “Opcional: habilitar marcadores de trabalho”.

Etapa 6. Confirmar o objeto **Job**

Você inicializou um objeto `Job` na Etapa 1. Você precisará concluir manualmente o ciclo de vida no final do script. Alguns recursos opcionais precisam que isso funcione corretamente. Esse trabalho é feito nos bastidores no AWS Glue Studio.

Nessa etapa, Chame o método `commit` no objeto `Job`.

Neste procedimento, você vai escrever o código a seguir. Esse código é uma parte do script de exemplo gerado.

```
job.commit()
```

Para confirmar o objeto **Job**

1. Caso ainda não tenha feito isso, execute as etapas opcionais descritas nas seções anteriores para incluir `transformation_ctx`.
2. Chame `commit`.

Opcional: habilitar marcadores de trabalho

Em cada etapa anterior, você recebeu instruções para definir parâmetros `transformation_ctx`. Isso está relacionado a um recurso chamado marcadores de trabalho.

Com marcadores de trabalho, você pode economizar tempo e dinheiro com trabalhos executados de maneira recorrente, em comparação com conjuntos de dados nos quais trabalhos anteriores podem ser facilmente rastreados. Os marcadores de tarefas monitoram o progresso de uma transformação do AWS Glue em um conjunto de dados de execuções anteriores. Ao rastrear onde as execuções anteriores terminaram, o AWS Glue pode limitar seu trabalho às linhas que não tenha processado antes. Para obter mais informações sobre marcadores de trabalho, consulte the section called “Rastrear dados processados usando marcadores de trabalho”.

Para ativar os marcadores de trabalho, primeiro adicione as declarações de `transformation_ctx` às nossas funções fornecidas, conforme descrito nos exemplos anteriores. O estado do marcador de trabalho é mantido em todas as execuções. Parâmetros de `transformation_ctx` são chaves usadas para acessar esse estado. Sozinhas, essas declarações não farão nada. Você também precisa ativar o recurso na configuração do seu trabalho.

Neste procedimento, você habilita os marcadores de trabalho usando o AWS Management Console.

Para definir marcadores de trabalho

1. Acesse a seção Job details (Detalhes de trabalho) do seu trabalho correspondente.
2. Defina Job bookmark (Marcador de trabalho) como Enable (Habilitar).

Etapa 7. Execute seu código como um trabalho

Nesta etapa, você executa seu trabalho para verificar se concluiu este tutorial com êxito. Isso é feito com o clique de um botão, como no editor visual do AWS Glue Studio.

Para executar seu código como um trabalho

1. Escolha Untitled job (Trabalho sem título) na barra de título para editar e definir o nome do seu trabalho.
2. Acesse a guia Job details (Detalhes do trabalho). Atribua um IAM Role (Perfil do IAM) ao seu trabalho. Você pode usar o criado pelo AWS CloudFormation modelo nos pré-requisitos do tutorial do AWS Glue Studio. Se tiver concluído esse tutorial, ele deve estar disponível como `AWS Glue StudioRole`.

3. Escolha Save (Salvar) para salvar seu script.
4. Escolha Run (Executar) para executar seu trabalho.
5. Navegue até a guia Runs (Execuções) para verificar se seu trabalho foi concluído.
6. Acesse *DOC-EXAMPLE-BUCKET*, o destino de `write_dynamic_frame.from_options`. Confirme se o resultado corresponde às suas expectativas.

Para obter mais informações sobre como configurar e gerenciar trabalhos, consulte the section called "Fornecer seus próprios scripts personalizados".

Mais informações

As bibliotecas e os métodos do Apache Spark estão disponíveis nos scripts AWS Glue. Você pode consultar a documentação do Spark para entender o que pode fazer com as bibliotecas incluídas. Para obter mais informações, consulte a seção de exemplos do repositório de origem do Spark.

AWS O Glue 2.0+ inclui várias bibliotecas Python comuns por padrão. Também há mecanismos para carregar suas próprias dependências em uma tarefa do AWS Glue em um ambiente Scala ou Python. Consulte the section called "Bibliotecas Python" para obter informações sobre dependências do Python.

Para obter mais exemplos de como usar os recursos do AWS Glue em Python, consulte. the section called "Exemplos de Python" Os trabalhos no Scala e no Python têm paridade de recursos; sendo assim, nossos exemplos do Python devem dar algumas ideias sobre como executar trabalhos semelhantes no Scala.

Programe scripts AWS Glue ETL em PySpark

Você pode encontrar exemplos e utilitários de código Python no repositório de AWS Glue amostras AWS Glue no site. GitHub

Usar o Python com o AWS Glue

AWS O Glue é compatível com uma extensão do dialeto PySpark Python para tarefas de extração, transformação e carregamento de scripts (ETL). Esta seção descreve como usar o Python em scripts de ETL e com a API do AWS Glue.

- Configurar Python para uso com o AWS Glue

- Chamar APIs do AWS Glue no Python
- Usar bibliotecas Python com o AWS Glue
- Exemplos de código Python do AWS Glue

AWS Glue PySpark extensões

AWS Gluecriou as seguintes extensões para o dialeto PySpark Python.

- Acessar parâmetros usando `getResolvedOptions`
- Tipos de extensão do PySpark
- Classe DynamicFrame
- Classe DynamicFrameCollection
- Classe DynamicFrameWriter
- DynamicFrameReader classe
- GlueContext classe

AWS Glue PySpark transforma-se

AWS Gluecriou as seguintes classes de transformação para usar em operações de PySpark ETL.

- Classe de base GlueTransform
- Classe ApplyMapping
- Classe DropFields
- Classe DropNullFields
- Classe ErrorsAsDynamicFrame
- Classe FillMissingValues
- Classe Filter
- Classe FindIncrementalMatches
- Classe FindMatches
- Classe FlatMap
- Classe Join
- Classe Map

- Classe MapToCollection
- mergeDynamicFrame
- Classe Relationalize
- Classe RenameField
- Classe ResolveChoice
- Classe SelectFields
- Classe SelectFromCollection
- Classe Spigot
- Classe SplitFields
- Classe SplitRows
- Classe Unbox
- Classe UnnestFrame

Configurar Python para uso com o AWS Glue

Use o Python para desenvolver seus scripts de ETL para trabalhos do Spark. As versões do Python compatíveis para trabalhos de ETL dependem da versão do AWS Glue do trabalho. Para obter mais informações sobre versões do AWS Glue, consulte Glue version job property.

Para configurar o sistema para usar o Python com o AWS Glue

Siga estas etapas para instalar o Python e invocar as APIs do AWS Glue.

1. Se você ainda não tem o Python instalado, faça download dele e instale-o acessando a página de download Python.org.
2. Instale a AWS Command Line Interface (AWS CLI) conforme descrito na Documentação da AWS CLI.

   A AWS CLI não é necessária para usar o Python. No entanto, instalar e configurar a CLI é uma maneira conveniente de configurar a AWS com as credenciais da sua conta e verificar se elas funcionam.
3. Instale o AWS SDK for Python (Boto 3), conforme descrito no Guia de início rápido do Boto3.

   As APIs de recursos do Boto 3 ainda não estão disponíveis no AWS Glue. Atualmente, apenas as APIs de cliente do Boto 3 podem ser usadas.

> Para obter mais informações sobre o Boto 3, consulte Conceitos básicos do AWS SDK for Python (Boto3).

É possível encontrar exemplos de código Python e utilitários para o AWS Glue no repositório de exemplos do AWS Glue no site do GitHub.

Chamar APIs do AWS Glue no Python

Observação: as APIs de recursos do Boto 3 ainda não estão disponíveis no AWS Glue. Atualmente, apenas as APIs de cliente do Boto 3 podem ser usadas.

Nomes de API do AWS Glue em Python

Os nomes de API do AWS Glue em Java e em outras linguagens de programação geralmente misturam letras maiúsculas e minúsculas. No entanto, quando chamados no Python, esses nomes genéricos são alterados para minúsculas, com partes do nome separadas por caracteres de sublinhado para deixá-los com uma aparência mais familiar para usuários do Python. Na documentação de referência da AWS Glue API, esses nomes do Python estão listados entre parênteses, após os nomes genéricos com letras minúsculas e maiúsculas misturadas.

Embora as letras dos nomes de API do AWS Glue sejam transformadas em minúsculas, seus nomes de parâmetros permanecem em letras maiúsculas. Lembre-se disso porque os parâmetros devem ser transmitidos pelo nome ao chamar APIs do AWS Glue, conforme descrito na seção a seguir.

Transmitir e acessar parâmetros de Python no AWS Glue

Nas chamadas do Python para APIs do AWS Glue, é melhor transmitir os parâmetros explicitamente por nome. Por exemplo:

```
job = glue.create_job(Name='sample', Role='Glue_DefaultRole',
                      Command={'Name': 'glueetl',
                               'ScriptLocation': 's3://my_script_bucket/scripts/
my_etl_script.py'})
```

O Python cria um dicionário das tuplas de nome/valor que você especifica como argumentos para um script de ETL em um Estrutura Job ou JobRun estrutura. Em seguida, o Boto 3 transmite para o AWS Glue no formato JSON por meio de uma chamada de API REST. Isso significa que você não pode confiar na ordem dos argumentos ao acessá-los no seu script.

Por exemplo, suponha que você esteja iniciando um `JobRun` em uma função de handler Lambda do Python e queira especificar vários parâmetros. Seu código será parecido com este:

```
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 acessar esses parâmetros de forma confiável no seu script de ETL, especifique-os pelo nome usando a função `getResolvedOptions` do AWS Glue e, em seguida, acesse-os no dicionário 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']
```

Se você quiser transmitir um argumento que seja uma string JSON aninhada, para preservar o valor do parâmetro à medida que ele é transmitido para o seu trabalho de ETL do AWS Glue, você deve codificar a string de parâmetro antes de iniciar a execução do trabalho e, em seguida, decodificar a string de parâmetro antes de fazer referência ao script de trabalho. Por exemplo, considere a seguinte string de argumento:

```
glue_client.start_job_run(JobName = "gluejobname", Arguments={
"--my_curly_braces_string": '{"a": {"b": {"c": [{"d": {"e": 42}}]}}}'
})
```

Para transmitir esse parâmetro corretamente, você deve codificar o argumento como uma string codificada em 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}")
...
```

Exemplo: criar e executar um trabalho

O exemplo a seguir mostra como chamar APIs do AWS Glue usando Python para criar e executar um trabalho de ETL.

Para criar e executar um trabalho

1. Crie uma instância do cliente 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. Crie um trabalho. Você precisa usar `glueetl` como o nome do comando de ETL, conforme mostrado no código a seguir:

   ```
   myJob = glue.create_job(Name='sample', Role='Glue_DefaultRole',
                           Command={'Name': 'glueetl',
   ```

```
                                    'ScriptLocation': 's3://my_script_bucket/
scripts/my_etl_script.py'})
```

3. Inicie uma nova execução do trabalho que você criou na etapa anterior:

```
myNewJobRun = glue.start_job_run(JobName=myJob['Name'])
```

4. Veja o status do trabalho:

```
status = glue.get_job_run(JobName=myJob['Name'], RunId=myNewJobRun['JobRunId'])
```

5. Imprima o estado atual da execução do trabalho:

```
print(status['JobRun']['JobRunState'])
```

Usar bibliotecas Python com o AWS Glue

O AWS Glue permite que você instale módulos e bibliotecas Python adicionais para uso com ETL do AWS Glue.

Tópicos

- Instalar módulos Python adicionais com pip no AWS Glue 2.0
- Incluindo arquivos Python com recursos nativos PySpark
- Scripts de programação que usam transformações visuais
- Módulos do Python já fornecidos no AWS Glue
- Compactar bibliotecas para inclusão
- Carregando bibliotecas Python nos cadernos do AWS Glue Studio
- Carregar bibliotecas Python em um endpoint de desenvolvimento
- Usando bibliotecas Python em um trabalho ou JobRun

Instalar módulos Python adicionais com pip no AWS Glue 2.0

O AWS Glue usa o Python Package Installer (pip3) para instalar módulos adicionais a serem usados pelo ETL do AWS Glue. Você pode usar o parâmetro `--additional-python-modules` com uma lista de módulos do Python separados por vírgulas para adicionar um novo módulo ou alterar a versão de um módulo existente. Você pode instalar distribuições personalizadas de uma biblioteca

carregando a distribuição para o Amazon S3 e incluindo o caminho para o objeto do Amazon S3 na sua lista de módulos.

Você pode transmitir opções adicionais para o pip3 com o parâmetro `--python-modules-installer-option`. Por exemplo, você pode transmitir `"--upgrade"` para atualizar os pacotes especificados por `"--additional-python-modules"`. Para ver mais exemplos, consulte Como criar módulos Python a partir de uma roda para cargas de trabalho de ETL do Spark usando o Glue 2.0. AWS

Se suas dependências do Python dependerem transitivamente de código compilado nativo, você pode se deparar com a seguinte limitação: O AWS Glue não oferece suporte à compilação de código nativo no ambiente de trabalho. No entanto, os trabalhos do AWS Glue são executados em um ambiente Amazon Linux 2. Talvez você possa fornecer suas dependências nativas em formato compilado por meio de um distribuível Wheel.

Por exemplo, para atualizar ou adicionar um novo módulo `scikit-learn`, use a seguinte chave-valor: `"--additional-python-modules", "scikit-learn==0.21.3"`.

Além disso, dentro da opção `--additional-python-modules`, você pode especificar um caminho do Amazon S3 para um módulo wheel do Python. Por exemplo: .

```
--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
```

Você especifica o `--additional-python-modules` no campo Parâmetros do trabalho do AWS Glue console ou alterando os argumentos do trabalho no AWS SDK. Para obter mais informações sobre como configurar parâmetros de trabalho, consulte the section called “Parâmetros de trabalho”.

Incluindo arquivos Python com recursos nativos PySpark

AWS Glueusa PySpark para incluir arquivos Python em trabalhos de ETL do AWS Glue. Recomenda-se usar `--additional-python-modules` para gerenciar suas dependências quando disponíveis. Você pode usar o parâmetro de trabalho `--extra-py-files` para incluir arquivos Python. As dependências devem ser hospedadas no Amazon S3, e o valor do argumento deve ser uma lista delimitada por vírgulas de caminhos do Amazon S3 sem espaços. Essa funcionalidade se comporta como o gerenciamento de dependências do Python que você usaria com o Spark. Para obter mais informações sobre o gerenciamento de dependências do Python no Spark, consulte a página Usando recursos PySpark nativos na documentação do Apache Spark. `--extra-py-files`é útil nos casos em que seu código adicional não está empacotado ou quando você

está migrando um programa Spark com uma cadeia de ferramentas existente para gerenciar dependências. Para que suas ferramentas de dependência sejam passíveis de manutenção, você precisará empacotar suas dependências antes de enviá-las.

Scripts de programação que usam transformações visuais

Ao criar uma tarefa do AWS Glue usando a interface visual do AWS Glue Studio, você pode transformar seus dados com nós de transformação de dados gerenciados e transformações visuais personalizadas. Para obter mais informações sobre nós de transformações de dados gerenciados, consulte the section called “Editar nós de transformação de dados gerenciados pelo AWS Glue”. Para obter mais informações sobre transformações visuais personalizadas, consulte the section called “ Transformações visuais personalizadas ”. Os scripts que usam transformações visuais só poderão ser gerados quando a Linguagem do trabalho estiver configurada para usar Python.

Ao gerar uma tarefa do AWS Glue usando transformações visuais, o AWS Glue Studio incluirá essas transformações no ambiente de execução usando o `--extra-py-files` parâmetro na configuração da tarefa. Para obter mais informações sobre parâmetros de trabalho, consulte the section called “Parâmetros de trabalho”. Ao fazer alterações em um script gerado ou em um ambiente de runtime, será necessário preservar essa configuração de trabalho para que seu script seja executado com êxito.

Módulos do Python já fornecidos no AWS Glue

Para alterar a versão desses módulos fornecidos, forneça novas versões com o parâmetro de trabalho `--additional-python-modules`.

AWS Glue version 2.0

O AWS Glue versão 2.0 inclui os seguintes módulos do Python prontos para 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 versão 3.0

O AWS Glue versão 3.0 inclui os seguintes módulos do Python prontos para 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 versão 4.0

O AWS Glue versão 4.0 inclui os seguintes módulos do Python prontos para 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

Compactar bibliotecas para inclusão

A menos que uma biblioteca esteja contida em um único arquivo `.py`, ela precisará ser compactada em uma pasta `.zip`. O diretório do pacote deve estar na raiz do arquivo e precisa conter um arquivo `__init__.py` para o pacote. Em seguida, o Python poderá importar o pacote normalmente.

Se a sua biblioteca contém um único módulo Python em um arquivo `.py`, você não precisará compactá-la em uma pasta `.zip`.

Carregando bibliotecas Python nos cadernos do AWS Glue Studio

Para especificar bibliotecas Python nos notebooks do AWS Glue Studio, consulte Instalação de módulos Python adicionais.

Carregar bibliotecas Python em um endpoint de desenvolvimento

Se você estiver usando diferentes conjuntos de bibliotecas para diferentes scripts de ETL, poderá configurar um endpoint de desenvolvimento separado para cada conjunto ou substituir os arquivos `.zip` da biblioteca carregados pelo seu endpoint de desenvolvimento sempre que os scripts são alternados.

Você pode usar o console para especificar um ou mais arquivos .zip da biblioteca para um endpoint de desenvolvimento ao criá-lo. Depois de atribuir um nome e uma função do IAM, escolha Script Libraries and job parameters (optional) (Bibliotecas de script e parâmetros de trabalho [opcional]) e insira o caminho completo do Amazon S3 para o arquivo `.zip` da sua biblioteca na caixa Python library path (Caminho da biblioteca Python). Por exemplo:

```
s3://bucket/prefix/site-packages.zip
```

Se quiser, você poderá especificar vários caminhos completos para arquivos, separando-os com vírgulas, mas sem espaços, da seguinte maneira:

```
s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/lib_X.zip
```

Se você atualizar esses arquivos `.zip` posteriormente, poderá usar o console para importá-los novamente para o seu endpoint de desenvolvimento. Navegue até o endpoint do desenvolvedor em questão, marque a caixa de seleção ao lado e escolha Update ETL libraries no menu Action.

De forma semelhante, você pode especificar arquivos de biblioteca usando as APIs do AWS Glue. Ao criar um endpoint de desenvolvimento chamando Ação CreateDevEndpoint (Python: create_dev_endpoint), você pode especificar um ou mais caminhos completos para bibliotecas no parâmetro `ExtraPythonLibsS3Path`. Essa chamada é semelhante à seguinte:

```
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")
```

Ao atualizar um endpoint de desenvolvimento, você também pode atualizar as bibliotecas que ele carrega usando um objeto DevEndpointCustomLibraries e definindo o parâmetro `UpdateEtlLibraries` como `True` ao chamar UpdateDevEndpoint (update_dev_endpoint).

Usando bibliotecas Python em um trabalho ou JobRun

Ao criar um novo trabalho no console, você pode especificar um ou mais arquivos .zip de biblioteca escolhendo a opção Script Libraries and job parameters (optional) (Bibliotecas de script e parâmetros de trabalho [opcional]) e inserindo os caminhos completos das bibliotecas do Amazon S3, da mesma maneira como você faria ao criar um endpoint de desenvolvimento:

```
s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/lib_X.zip
```

Se você estiver chamando CreateJob (criar_emprego), poderá especificar um ou mais caminhos completos para bibliotecas padrão usando o parâmetro padrão `--extra-py-files`, da seguinte maneira:

```
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'})
```

Então, ao iniciar um JobRun, você pode substituir a configuração padrão da biblioteca por uma diferente:

```
runId = glue.start_job_run(JobName='sampleJob',
                           Arguments={'--extra-py-files': 's3://bucket/prefix/
lib_B.zip'})
```

Exemplos de código Python do AWS Glue

- Exemplo de código: juntar e relacionar dados
- Exemplo de código: preparo de dados usando ResolveChoice, Lambda e ApplyMapping

Exemplo de código: juntar e relacionar dados

Este exemplo usa um conjunto de dados obtidos por download no site http://everypolitician.org/ para o bucket `sample-dataset` no Amazon Simple Storage Service (Amazon S3): `s3://awsglue-datasets/examples/us-legislators/all`. O conjunto de dados contém dados no formato JSON sobre os legisladores dos Estados Unidos e os assentos que eles ocuparam na Câmara dos Deputados e no Senado Americano e foi ligeiramente modificado e disponibilizado em um bucket público do Amazon S3 para a finalidade deste tutorial.

Você pode encontrar o código-fonte para este exemplo no arquivo `join_and_relationalize.py` no repositório de exemplos do AWS Glue no site do GitHub.

Usando esses dados, você realizará os procedimentos a seguir:

- Use um crawler do AWS Glue para classificar objetos armazenados em um bucket público do Amazon S3 e salvar seus esquemas no catálogo de dados do Glue da AWS.
- Examine os metadados e esquemas da tabela resultantes do rastreamento.
- Escreva um script Python de extração, transferência e carregamento (ETL) que usa os metadados no Data Catalog para fazer o seguinte:
  - Juntar dados nos diferentes arquivos de fonte em uma única tabela de dados (ou seja, desnormalizar os dados).

- Filtrar a tabela reunida em tabelas separadas por tipo de legislador.
- Gravar dados resultantes em arquivos Apache Parquet separados para análise posterior.

A forma preferida de depurar scripts do Python ou do PySpark durante a execução na AWS é usar cadernos no AWS Glue Studio.

Etapa 1: crawling de dados no bucket do Amazon S3

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Siga as etapas em Configurar um crawler para criar um novo crawler que possa fazer crawling no conjunto de dados `s3://awsglue-datasets/examples/us-legislators/all` em um banco de dados nomeado `legislators` no AWS Glue Data Catalog. Os dados de exemplo já estão nesse bucket público do Amazon S3.
3. Execute o novo crawler e, em seguida, verifique o banco de dados `legislators`.

   O crawler cria as seguintes tabelas de metadados:

   - `persons_json`
   - `memberships_json`
   - `organizations_json`
   - `events_json`
   - `areas_json`
   - `countries_r_json`

   Esta é uma coleção seminormalizada de tabelas que contêm legisladores e suas histórias.

Etapa 2: adicionar o script de boilerplate ao caderno do endpoint de desenvolvimento

Cole o seguinte script de boilerplate no caderno do endpoint de desenvolvimento para importar as bibliotecas do AWS Glue necessárias e configure um 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())
```

Etapa 3: examinar os esquemas dos dados no Data Catalog

Em seguida, você pode facilmente criar e examinar um DynamicFrame a partir do AWS Glue Data Catalog e examinar os esquemas dos dados. Por exemplo, para ver o esquema da tabela `persons_json`, adicione o seguinte ao seu caderno:

```
persons = glueContext.create_dynamic_frame.from_catalog(
             database="legislators",
             table_name="persons_json")
print "Count: ", persons.count()
persons.printSchema()
```

Veja a seguir a saída das chamadas de impressão:

```
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 pessoa na tabela é membro de algum órgão do Congresso dos EUA.

Para visualizar o esquema da tabela `memberships_json`, digite o seguinte:

```
memberships = glueContext.create_dynamic_frame.from_catalog(
                 database="legislators",
                 table_name="memberships_json")
print "Count: ", memberships.count()
memberships.printSchema()
```

A saída é a seguinte:

```
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
```

As `organizations` são partidos e as duas Câmaras do Congresso, o Senado e a Câmara dos Deputados. Para visualizar o esquema da tabela `organizations_json`, digite o seguinte:

```
orgs = glueContext.create_dynamic_frame.from_catalog(
```

```
           database="legislators",
           table_name="organizations_json")
print "Count: ", orgs.count()
orgs.printSchema()
```

A saída é a seguinte:

```
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
```

Etapa 4: filtrar os dados

Em seguida, mantenha apenas os campos desejados e renomeie `id` como `org_id`. O conjunto de dados é pequeno o suficiente para que você possa vê-lo por completo.

O `toDF()` converte um `DynamicFrame` em um `DataFrame` do Apache Spark, para que você possa aplicar as transformações já existentes no Apache Spark SQL:

```
orgs = orgs.drop_fields(['other_names',
                         'identifiers']).rename_field(
                             'id', 'org_id').rename_field(
                                'name', 'org_name')
```

```
orgs.toDF().show()
```

Veja a saída a seguir:

```
+-------------+--------------------+--------------------+--------------------+-----
+-----------+--------------------+
|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|
+-------------+--------------------+--------------------+--------------------+-----
+-----------+--------------------+
```

Digite o seguinte para visualizar `organizations` que aparecem em `memberships`:

```
memberships.select_fields(['organization_id']).toDF().distinct().show()
```

Veja a saída a seguir:

```
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

Etapa 5: integrar tudo

Agora, use o AWS Glue para juntar essas tabelas relacionais e criar uma tabela de histórico completa do legislador `memberships` e suas correspondentes `organizations`.

1. Primeiro, junte `persons` e `memberships` em `id` e `person_id`.
2. Em seguida, junte o resultado com `orgs` em `org_id` e `organization_id`.
3. Coloque os campos redundantes `person_id` e `org_id`.

Você pode fazer todas essas operações em uma linha de código (extensa):

```
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()
```

A saída é a seguinte:

```
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
```

Agora você tem a tabela final que pode usar para análise. Você pode gravar em um formato compacto e eficiente para análise (ou seja, Parquet) que permita a execução de SQL no AWS Glue, no Amazon Athena ou no Amazon Redshift Spectrum.

A seguinte chamada grava a tabela em vários arquivos para suportar leituras paralelas rápidas ao fazer a análise posteriormente:

```
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 colocar todos os dados do histórico em um único arquivo, você precisa convertê-los em um quadro de dados, reparticioná-los e gravá-los:

```
s_history = l_history.toDF().repartition(1)
s_history.write.parquet('s3://glue-sample-target/output-dir/legislator_single')
```

Se preferir, você pode separá-los por "Senado" e "Câmara":

```
l_history.toDF().write.parquet('s3://glue-sample-target/output-dir/legislator_part',
                               partitionBy=['org_name'])
```

Etapa 6: transformar os dados para bancos de dados relacionais

O AWS Glue facilita a gravação dos dados em bancos de dados relacionais, como o Amazon RedShift, mesmo com dados semiestruturados. Ele oferece uma transformação `relationalize`, que nivela `DynamicFrames` independentemente da complexidade dos objetos no quadro.

Usando o `l_history` `DynamicFrame` neste exemplo, transmita o nome da tabela raiz (`hist_root`) e um caminho de trabalho temporário para `relationalize`. Isso retorna um `DynamicFrameCollection`. Você pode então os nomes dos `DynamicFrames` nessa coleção:

```
dfc = l_history.relationalize("hist_root", "s3://glue-sample-target/temp-dir/")
dfc.keys()
```

Veja a seguir a saída da chamada `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` quebrou a tabela de histórico em seis novas tabelas: uma tabela raiz que contém um registro para cada objeto no `DynamicFrame`, e tabelas auxiliares para as matrizes. O gerenciamento de matrizes em bancos de dados relacionais geralmente é pouco satisfatório, principalmente porque essas matrizes ficam grandes. Separar as matrizes em diferentes tabelas torna as consultas muito mais rápidas.

Em seguida, verifique a separação 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()
```

Veja a seguir a saída da chamada `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|
+---+-----+-----------------------+-------------------------+
```

O campo `contact_details` era uma matriz de estruturas no `DynamicFrame` original. Cada elemento dessas matrizes é uma linha separada na tabela auxiliar, indexada por `index`. Aqui, `id` é uma chave externa na tabela `hist_root` com a chave `contact_details`:

```
dfc.select('hist_root').toDF().where(
    "contact_details = 10 or contact_details = 75").select(
       ['id', 'given_name', 'family_name', 'contact_details']).show()
```

Esta é a saída:

```
+--------------------+----------+-----------+---------------+
|                  id|given_name|family_name|contact_details|
+--------------------+----------+-----------+---------------+
|f4fc30ee-7b42-432...|      Mike|       Ross|             10|
|e3c60f34-7d1b-4c0...|   Shelley|     Capito|             75|
+--------------------+----------+-----------+---------------+
```

Nestes comandos, as expressões `toDF()` e `where` são usadas para filtrar as linhas que você quer ver.

Portanto, ao juntar a tabela `hist_root` e as tabelas auxiliares, você pode fazer o seguinte:

- Carregar dados em bancos de dados sem suporte de matriz.
- Consultar cada item individual em uma matriz usando o SQL.

Armazene e acesse com segurança suas credenciais do Amazon Redshift com uma conexão do AWS Glue. Para obter informações sobre como criar suas próprias conexões, consulte Conectar a dados.

Agora está tudo pronto para gravar seus dados em uma conexão alternando por um `DynamicFrames` de cada vez:

```
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)
```

Suas configurações de conexão serão diferentes de acordo com o tipo de banco de dados relacional:

- Para obter instruções sobre como gravar no Amazon Redshift, consulte the section called "Conexões do Redshift".
- Para outros bancos de dados, consulte Tipos e opções de conexão para ETL no AWS Glue para Spark.

Conclusão

De modo geral, o AWS Glue é muito flexível. Com ele, você usa algumas linhas de código para escrever o que normalmente levaria dias. Você pode encontrar todos os scripts de ETL de fonte para destino no arquivo Python `join_and_relationalize.py` nos exemplos do AWS Glue no GitHub.

Exemplo de código: preparo de dados usando ResolveChoice, Lambda e ApplyMapping

O conjunto de dados que é usado neste exemplo consiste nos dados de pagamento de um provedor da Medicare obtidos por download em dois conjuntos de dados do Data.CMS.gov: "Inpatient Prospective Payment System Provider Summary for the Top 100 Diagnosis-Related Groups - FY2011" e "Inpatient Charge Data FY 2011". Depois de fazer download dos dados, nós os modificamos para apresentar alguns registros errados ao final do arquivo. Esse arquivo modificado está localizado em um bucket público do Amazon S3 em `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv`.

Você pode encontrar o código-fonte para este exemplo no arquivo `data_cleaning_and_lambda.py` nos exemplos do AWS Glue do repositório GitHub.

A forma preferida de depurar scripts do Python ou do PySpark durante a execução na AWS é usar cadernos no AWS Glue Studio.

Etapa 1: crawling de dados no bucket do Amazon S3

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Seguindo o processo descrito em Configurar um crawler, crie um novo crawler que possa fazer crawling no arquivo `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv` e colocar os metadados resultantes em um banco de dados chamado `payments` no AWS Glue Data Catalog.
3. Execute o novo crawler e, em seguida, verifique o banco de dados `payments`. Você deve descobrir que o crawler criou uma tabela de metadados nomeada `medicare` no banco de dados depois de ler o início do arquivo para determinar seu formato e delimitador.

O esquema da nova tabela `medicare` será o seguinte:

```
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
```

Etapa 2: adicionar o script de boilerplate ao caderno do endpoint de desenvolvimento

Cole o seguinte script de boilerplate no caderno do endpoint de desenvolvimento para importar as bibliotecas do AWS Glue necessárias e configure um 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())
```

Etapa 3: comparar diferentes análises de esquema

Em seguida, você poderá ver se o esquema que foi reconhecido por um `DataFrame` do Apache Spark é igual àquele que seu crawler do AWS Glue gravou. Execute este código:

```

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()
```

Veja a seguir a saída da chamada `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)
```

Em seguida, veja o esquema gerado por um AWS Glue `DynamicFrame`:

```
medicare_dynamicframe = glueContext.create_dynamic_frame.from_catalog(
       database = "payments",
       table_name = "medicare")
medicare_dynamicframe.printSchema()
```

A saída de `printSchema` é a seguinte:

```
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
```

O `DynamicFrame` gera um esquema em que `provider id` pode ser um tipo `long` ou `string`. O esquema `DataFrame` lista `Provider Id` como um tipo `string`, e o Data Catalog lista `provider id` como um tipo `bigint`.

Qual está correto? Existem dois registros no final do arquivo (de 160 mil registros) com valores `string` na coluna em questão. Estes são os registros errados que foram apresentados para ilustrar um problema.

Para resolver esse tipo de problema, o AWS Glue `DynamicFrame` apresenta o conceito de um tipo de escolha. Neste caso, o `DynamicFrame` mostra que os valores `long` e `string` podem ser exibidos na coluna. O crawler do AWS Glue não obteve os valores `string` porque considerou apenas um prefixo de 2 MB dos dados. O `DataFrame` do Apache Spark considerou todo o conjunto de dados, mas foi forçado a atribuir o tipo mais geral à coluna, ou seja, `string`. Na verdade, o Spark recorre frequentemente ao caso mais geral quando há tipos complexos ou variações com os quais ele não está familiarizado.

Para consultar a coluna `provider id`, resolva primeiro o tipo de escolha. Você pode usar o método de transformação `resolveChoice` no seu `DynamicFrame` para converter os valores `string` para `long` com uma opções `cast:long`:

```
medicare_res = medicare_dynamicframe.resolveChoice(specs = [('provider
 id','cast:long')])
medicare_res.printSchema()
```

A saída de `printSchema` agora é:

```
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
```

Em que o valor era uma `string` que não podia ser convertida, o AWS Glue inseriu `null`.

Outra opção é converter o tipo de escolha para `struct`, que mantém valores de ambos os tipos.

Em seguida, veja as linhas que eram anômalas:

```
medicare_res.toDF().where("'provider id' is NULL").show()
```

Você verá o seguinte:

```
+--------------------+-----------+---------------+----------------------+-------------
+--------------+-----------------+-------------------------------------+---------------
+----------------------+---------------------+------------------------+
|      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|
+--------------------+-----------+---------------+----------------------+-------------
+--------------+-----------------+-------------------------------------+---------------
+----------------------+---------------------+------------------------+
```

Agora, remova os dois registros malformados, da seguinte forma:

```
medicare_dataframe = medicare_res.toDF()
medicare_dataframe = medicare_dataframe.where("'provider id' is NOT NULL")
```

Etapa 4: mapear os dados e usar as funções Lambda do Apache Spark

O AWS Glue ainda não oferece suporte diretamente às funções Lambda, também conhecidas como funções definidas pelo usuário. No entanto, você sempre pode converter um `DynamicFrame` de e para um `DataFrame` do Apache Spark para utilizar as funcionalidades do Spark e os recursos especiais de `DynamicFrames`.

Em seguida, transforme as informações de pagamento em números para que os mecanismos de análise, como Amazon RedShift ou Amazon Athena, possam fazer seus respectivas análises mais rapidamente:

```
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()
```

A saída da chamada de `show` é a seguinte:

```
+--------+-------+-------+
|     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
```

Estas são todas as strings nos dados. Podemos usar o eficiente método de transformação `apply_mapping` para soltar, renomear, converter e aninhar os dados de modo que outros idiomas e sistemas de programação de dados possam acessá-los facilmente:

```
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()
```

A saída de `printSchema` é a seguinte:

```
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
```

Direcionando os dados de volta para um `DataFrame` do Spark, algo parecido com isto será exibido:

```
medicare_nest_dyf.toDF().show()
```

A saída é a seguinte:

```
+--------------------+--------------------+--------------+--------------------+
|                 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
```

Etapa 5: gravar os dados no Apache Parquet

O AWS Glue facilita a escrita dos dados em um formato como o Apache Parquet, que os bancos de dados relacionais possam efetivamente consumir:

```
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")
```

Referência de extensões PySpark do AWS Glue

O AWS Glue criou as seguintes extensões para o dialeto PySpark Python.

- Acessar parâmetros usando `getResolvedOptions`
- Tipos de extensão do PySpark
- Classe DynamicFrame
- Classe DynamicFrameCollection
- Classe DynamicFrameWriter
- DynamicFrameReader classe
- GlueContext classe

Acessar parâmetros usando **`getResolvedOptions`**

A função do utilitário AWS Glue `getResolvedOptions(args, options)` concede a você acesso aos argumentos transmitidos para o seu script quando você executa um trabalho. Para usar essa função, comece importando-a do módulo `utils` do AWS Glue junto com o `sys`:

```
import sys
from awsglue.utils import getResolvedOptions
```

`getResolvedOptions(args, options)`

- `args` – A lista de argumentos contida em `sys.argv`.
- `options` – Uma matriz Python dos nomes de argumentos que você quer recuperar.

Example Recuperar argumentos transmitidos para um JobRun

Suponha que você tenha criado um JobRun em um script, talvez dentro de uma função do 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 os argumentos transmitidos, você pode usar a função `getResolvedOptions`, da seguinte maneira:

```
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']
```

Observe que cada um dos argumentos são definidos como começando com dois hifens e, então, referenciados no script sem os hifens. Os argumentos usam apenas sublinhados, não hífens. Seus argumentos precisam seguir essa convenção para serem resolvidos.

Tipos de extensão do PySpark

Os tipos usados pelas extensões de PySpark do AWS Glue.

DataType

A classe base para outros tipos do AWS Glue.

`__init__(properties={})`

- `properties` – Propriedades do tipo de dados (opcional).

`typeName(cls)`

Retorna o tipo da classe do AWS Glue (ou seja, o nome da classe com "Tipo" removido do final).

- `cls` – Uma instância de classe do AWS Glue derivada de `DataType`.

`jsonValue( )`

Retorna um objeto JSON que contém o tipo de dados e as propriedades da classe:

```
{
  "dataType": typeName,
  "properties": properties
}
```

Derivativos AtomicType e Simple

Herda e estende a classe DataType e serve como a classe base para todos os tipos de dados atômicos do AWS Glue.

`fromJsonValue(cls, json_value)`

Inicializa uma instância de classe com valores de um objeto JSON.

- `cls` – Uma instância de classe de tipo do AWS Glue a ser inicializada.
- `json_value` – O objeto JSON de onde os pares de valor-chave serão carregados.

Os seguintes tipos são derivados simples da classe AtomicType:

- `BinaryType` – Dados binários.
- `BooleanType` – Valores boolianos.
- `ByteType` – Um valor de bytes.
- `DateType` – Um valor de data e hora.
- `DoubleType` – Um valor com ponto flutuante duplo.
- `IntegerType` – Um valor inteiro.
- `LongType` – Um valor inteiro extenso.
- `NullType` – Um valor nulo.
- `ShortType` – Um valor inteiro curto.
- `StringType` – Uma string de texto.

- `TimestampType` – Um valor de data e hora (geralmente em segundos, a partir de 01/01/1970).
- `UnknownType` – Um valor de tipo não identificado.

DecimalType(AtomicType)

Herda e estende a classe AtomicType para representar um número decimal (um número expresso em dígitos decimais, em oposição aos números binários de base 2).

`__init__(precision=10, scale=2, properties={})`

- `precision` - O número de dígitos no número decimal (opcional; o padrão é 10).
- `scale` – O número de dígitos à direita da vírgula decimal (opcional, o padrão é 2).
- `properties` – As propriedades do número decimal (opcional).

EnumType(AtomicType)

Herda e estende a classe AtomicType para representar uma enumeração de opções válidas.

`__init__(options)`

- `options` – Uma lista de opções sendo enumeradas.

tipos de coleção

- ArrayType(DataType)
- ChoiceType(DataType)
- MapType(DataType)
- Field(Object)
- StructType(DataType)
- EntityType(DataType)

ArrayType(DataType)

`__init__(elementType=UnknownType(), properties={})`

- `elementType` – O tipo dos elementos na matriz (opcional, o padrão é UnknownType).

- `properties` – Propriedades da matriz (opcional).

ChoiceType(DataType)

`__init__(choices=[], properties={})`

- `choices` – Uma lista das opções possíveis (opcional).
- `properties` – As propriedades dessas opções (opcional).

`add(new_choice)`

Adiciona uma nova opção à lista de opções possíveis.

- `new_choice` – A opção a ser adicionada à lista de opções possíveis.

`merge(new_choices)`

Combina uma lista de novas opções com a lista de opções existentes.

- `new_choices` – Uma lista de novas opções a ser combinada com a lista de opções existentes.

MapType(DataType)

`__init__(valueType=UnknownType, properties={})`

- `valueType` – O tipo dos valores no mapa (opcional, o padrão é UnknownType).
- `properties` – Propriedades do mapa (opcional).

Field(Object)

Cria um objeto de campo de um objeto que deriva de DataType.

`__init__(name, dataType, properties={})`

- `name` – O nome a ser atribuído ao campo.
- `dataType` – O objeto a ser criado a partir de um campo.

- `properties` – Propriedades do campo (opcional).

StructType(DataType)

Define uma estrutura de dados (`struct`).

`__init__(fields=[], properties={})`

- `fields` – Uma lista dos campos (do tipo `Field`) a ser incluída na estrutura (opcional).
- `properties` – Propriedades da estrutura (opcional).

`add(field)`

- `field` – Um objeto do tipo `Field` a ser adicionado à estrutura.

`hasField(field)`

Retorna `True` se essa estrutura tiver um campo com o mesmo nome ou `False` caso contrário.

- `field` – O nome de um campo ou um objeto do tipo `Field` cujo nome está sendo usado.

`getField(field)`

- `field`: um nome de um campo ou um objeto do tipo `Field` cujo nome está sendo usado. Se a estrutura tiver um campo com o mesmo, ele será retornado.

EntityType(DataType)

`__init__(entity, base_type, properties)`

Esta classe ainda não foi implementada.

outros tipos

- DataSource(object)
- DataSink(object)

DataSource(object)

`__init__(j_source, sql_ctx, name)`

- `j_source` – A fonte de dados.
- `sql_ctx` – O contexto SQL.
- `name` – O nome da fonte de dados.

`setFormat(format, **options)`

- `format` – O formato a ser configurado para a fonte de dados.
- `options` – Uma coleção de opções a ser configurada para a fonte de dados. Para obter mais informações sobre essas opções de formato, consulte the section called “Opções de formato de dados”.

`getFrame()`

Retorna um `DynamicFrame` para a fonte de dados.

DataSink(object)

`__init__(j_sink, sql_ctx)`

- `j_sink` – O depósito a ser criado.
- `sql_ctx` – O contexto SQL para o depósito de dados.

`setFormat(format, **options)`

- `format` – O formato a ser configurado para o depósito de dados.
- `options` – Uma coleção de opções a ser configurada para o depósito de dados. Para obter mais informações sobre essas opções de formato, consulte the section called “Opções de formato de dados”.

setAccumulableSize(size)

- size – O tamanho acumulável a ser configurado, em bytes.

writeFrame(dynamic_frame, info="")

- dynamic_frame – O DynamicFrame a ser escrito.
- info – Informações sobre o DynamicFrame (opcional).

write(dynamic_frame_or_dfc, info="")

Escreve DynamicFrame ou DynamicFrameCollection.

- dynamic_frame_or_dfc – Um objeto DynamicFrame ou DynamicFrameCollection a ser escrito.
- info – Informações sobre DynamicFrame ou DynamicFrames a serem escritas (opcional).

Classe DynamicFrame

Uma das principais abstrações no Apache Spark é o DataFrame do SparkSQL, que é semelhante à construção DataFrame encontrada no R e Pandas. Um DataFrame é semelhante a uma tabela e é compatível com operações de estilo funcional (mapear/reduzir/filtrar etc.) e operações SQL (select, project, aggregate).

DataFrames são potentes e amplamente utilizados, mas têm limitações em relação às operações de extração, transformação e carregamento (ETL). Mais importante, eles exigem que um esquema seja especificado antes que qualquer dado seja carregado. O SparkSQL soluciona essa questão fazendo duas transmissões sobre dados: a primeira para inferir o esquema e a segunda para carregar os dados. No entanto, essa inferência é limitada e não corrige a desorganização de dados. Por exemplo, o mesmo campo pode ser de um tipo diferente em registros diferentes. O Apache Spark muitas vezes desiste e relata o tipo como string usando o texto do campo original. Ele pode estar incorreto, e convém ter controle mais preciso sobre como as discrepâncias do esquema são resolvidas. Para grandes conjuntos de dados, uma transmissão adicional sobre os dados de origem pode ser proibitivamente dispendiosa.

Para resolver essas limitações, o AWS Glue apresenta o DynamicFrame. Um DynamicFrame é semelhante a DataFrame, mas cada registro se autodescrevente, então nenhum esquema é

necessário inicialmente. Em vez disso, o AWS Glue calcula um esquema instantaneamente quando necessário e codifica explicitamente inconsistências de esquema usando um tipo de escolha (ou união). Você pode resolver essas inconsistências para tornar seus conjuntos de dados compatíveis com armazenamentos de dados que exigem um esquema fixo.

Da mesma forma, um `DynamicRecord` representa um registro lógico em um `DynamicFrame`. Ele é igual a uma linha em um `DataFrame` do Spark, exceto pelo fato de que ele pode se autodescrever e ser usado para dados que não estão em conformidade com um esquema fixo. Ao usar o AWS Glue com o PySpark, você normalmente não manipula `DynamicRecords` independentes. Em vez disso, você transformará o conjunto de dados junto por meio de seu `DynamicFrame`.

Você pode converter `DynamicFrames` de e para `DataFrames` depois de resolver as inconsistências de esquema.

— construção —

- __init__
- fromDF
- toDF

__init__

`__init__(jdf, glue_ctx, name)`

- `jdf` – Uma referência ao quadro de dados na Java Virtual Machine (JVM).
- `glue_ctx` – Um objeto GlueContext classe.
- `name` – Uma string de nome opcional, vazia por padrão.

fromDF

`fromDF(dataframe, glue_ctx, name)`

Converte um `DataFrame` em um `DynamicFrame`, transformando campos `DataFrame` em campos `DynamicRecord`. Retorna um novo `DynamicFrame`.

Um `DynamicRecord` representa um registro lógico em um `DynamicFrame`. Ele é semelhante a uma linha em um `DataFrame` do Spark, exceto pelo fato de que ele pode se autodescrever e ser usado para dados que não estão em conformidade com um esquema fixo.

Essa função espera que as colunas com nomes duplicados em seu `DataFrame` já tenham sido resolvidas.

- `dataframe` – O `DataFrame` Apache Spark SQL a ser convertido (obrigatório).
- `glue_ctx` – O objeto GlueContext classe que especifica o contexto para essa transformação (obrigatório).
- `name`: o nome do `DynamicFrame` resultante (opcional desde o AWS Glue 3.0).

toDF

`toDF(options)`

Converte um `DynamicFrame` a um `DataFrame` do Apache Spark, transformando campos `DynamicRecords` em `DataFrame`. Retorna um novo `DataFrame`.

Um `DynamicRecord` representa um registro lógico em um `DynamicFrame`. Ele é semelhante a uma linha em um `DataFrame` do Spark, exceto pelo fato de que ele pode se autodescrever e ser usado para dados que não estão em conformidade com um esquema fixo.

- `options`: uma lista de opções. Especifique o tipo de destino se você escolher o tipo de ação `Project` e `Cast`. Os exemplos incluem.

```
>>>toDF([ResolveOption("a.b.c", "KeepAsStruct")])
>>>toDF([ResolveOption("a.b.c", "Project", DoubleType())])
```

— informações —

- contagem
- Esquema
- printSchema
- show
- repartição
- coalesce

contagem

`count( )` – Retorna o número de linhas no `DataFrame` subjacente.

Esquema

`schema( )` – Retorna o esquema deste `DynamicFrame` ou, se não estiver disponível, o esquema do `DataFrame` subjacente.

Para obter mais informações sobre os tipos de `DynamicFrame` que compõem esse esquema, consulte the section called “Tipos”.

printSchema

`printSchema( )` – Imprime o esquema do `DataFrame` subjacente.

show

`show(num_rows)` – Imprime um número de linhas especificado do `DataFrame` subjacente.

repartição

`repartition(numPartitions)`: retorna um novo `DynamicFrame` com `numPartitions` partições.

coalesce

`coalesce(numPartitions)`: retorna um novo `DynamicFrame` com `numPartitions` partições.

— transformações —

- apply_mapping
- drop_fields
- filtrar
- join
- mapear
- mergeDynamicFrame
- relationalize
- rename_field
- resolveChoice
- select_fields

- spigot
- split_fields
- split_rows
- unbox
- the section called “união”
- unnest
- unnest_ddb_json
- write

apply_mapping

`apply_mapping(mappings, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Aplica um mapeamento declarativo a um `DynamicFrame` e retorna um novo `DynamicFrame` com esses mapeamentos aplicados aos campos que você especificar. Os campos não especificados são omitidos do novo `DynamicFrame`.

- `mappings`: uma lista de tuplas de mapeamento (obrigatório). Cada uma é composta por: coluna de fonte, tipo de fonte, coluna de destino, tipo de destino.

  Se houver um ponto "`.`" no nome da coluna de origem, será necessário colocá-lo entre crases "```". Por exemplo, para mapear `this.old.name` (string) para `thisNewName`, você pode usar a seguinte tupla:

  ```
  ("`this.old.name`", "string", "thisNewName", "string")
  ```

- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

Exemplo: usar apply_mapping para renomear campos e alterar os tipos de campo

O exemplo de código a seguir mostra como usar o método `apply_mapping` para renomear campos selecionados e alterar os tipos de campo.

> **Note**
>
> Para acessar o conjunto de dados usado neste exemplo, consulte Exemplo de código: juntar e relacionar dados e siga as instruções em Etapa 1: crawling de dados no bucket do 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()
```

Saída

```
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)`

Chama a transformação Classe FlatMap para remover campos de um `DynamicFrame`. Retorna um novo `DynamicFrame` com os campos especificados descartados.

- `paths`: uma lista de cadeias de caracteres. Cada um contém o caminho completo para um nó de campo que você deseja descartar. É possível usar notação de pontos para especificar campos aninhados. Por exemplo, se campo `first` for secundário do campo `name` na árvore, especifique `"name.first"` para o caminho.

  Se houver um literal `.` no nome de um nó de campo, será necessário colocá-lo entre crases (`` ` ``).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

Exemplo: usar drop_fields para remover campos de um **`DynamicFrame`**

Este código de exemplo usa o método `drop_fields` para remover campos aninhados e de nível superior selecionados de um `DynamicFrame`.

Exemplo de conjunto de dados

O exemplo usa o seguinte conjunto de dados que é representado pela tabela `EXAMPLE-FRIENDS-DATA` no 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 exemplo

```
# 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()
```

Saída

```
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
```

filtrar

`filter(f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Retorna um novo `DynamicFrame` que contém todos os `DynamicRecords` no `DynamicFrame` de entrada que satisfazem uma função `f` predicada especificada.

- `f`: a função predicada a ser aplicada a `DynamicFrame`. A função precisa ter um `DynamicRecord` como um argumento e retornar True, se `DynamicRecord` atender aos requisitos de filtro, ou False, caso contrário (obrigatório).

  Um `DynamicRecord` representa um registro lógico em um `DynamicFrame`. É semelhante a uma linha em um `DataFrame` do Spark, exceto pelo fato de que pode se autodescrever e ser usado para dados que não estejam em conformidade com um esquema fixo.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

Exemplo: usar filter para obter uma seleção filtrada de campos

Este exemplo usa o método `filter` para criar um novo `DynamicFrame` que inclui uma seleção filtrada de outros campos do `DynamicFrame`.

Assim como o método `map`, `filter` usa uma função como um argumento que é aplicado a cada registro no `DynamicFrame` original. A função usa um registro como entrada e retorna um valor booleano. Se o valor de retorno for true, o registro será incluído no `DynamicFrame` resultante. Se for false, o registro será omitido.

> **Note**
>
> Para acessar o conjunto de dados usado neste exemplo, consulte Exemplo de código: preparo de dados usando ResolveChoice, Lambda e ApplyMapping e siga as instruções em Etapa 1: crawling de dados no bucket do 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())
```

Saída

```
Unfiltered record count:  163065
Filtered record count:   564
```

join

`join(paths1, paths2, frame2, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Executa uma junção de igualdade com outro `DynamicFrame` e retorna o `DynamicFrame` resultante.

- `paths1` – Uma lista das chaves neste quadro para realizar a junção.
- `paths2` – Uma lista das chaves em outro quadro para realizar a junção.
- `frame2` - O outro `DynamicFrame` para realizar a junção.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

Exemplo: usar join para combinar **`DynamicFrames`**

Este exemplo usa o método `join` para realizar uma junção em três `DynamicFrames`. O AWS Glue executa a junção com base nas chaves de campo que você fornece. O `DynamicFrame` resultante contém linhas dos dois quadros originais em que as chaves especificadas correspondem.

A transformação do `join` mantém todos os campos intactos. Isso significa que os campos que você especificar para correspondência serão exibidos no DynamicFrame resultante, mesmo que sejam redundantes e contenham as mesmas chaves. Neste exemplo, usamos `drop_fields` para remover essas chaves redundantes após a junção.

> **Note**
>
> Para acessar o conjunto de dados usado neste exemplo, consulte Exemplo de código: juntar e relacionar dados e siga as instruções em Etapa 1: crawling de dados no bucket do 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()
```

Saída

```
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
```

mapear

`map(f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Retorna um novo `DynamicFrame` que resulta da aplicação da função de mapeamento especificada a todos os registros no `DynamicFrame` original.

- `f`: a função de mapeamento a ser aplicada a todos os registros em `DynamicFrame`. A função precisa levar um `DynamicRecord` como um argumento e retornar um novo `DynamicRecord` (obrigatório).

  Um `DynamicRecord` representa um registro lógico em um `DynamicFrame`. É semelhante a uma linha em um `DataFrame` do Apache Spark, exceto pelo fato de que pode se autodescrever e ser usado para dados que não estejam em conformidade com um esquema fixo.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info`: uma cadeira de caracteres que é associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

Exemplo: usar map para aplicar uma função a cada registro em um **`DynamicFrame`**

Este exemplo mostra como usar o método `map` para aplicar uma função a cada registro de um `DynamicFrame`. Especificamente, este exemplo aplica uma função chamada `MergeAddress` para cada registro para mesclar vários campos de endereço em um único tipo `struct`.

> **Note**
>
> Para acessar o conjunto de dados usado neste exemplo, consulte Exemplo de código: preparo de dados usando ResolveChoice, Lambda e ApplyMapping e siga as instruções em Etapa 1: crawling de dados no bucket do 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()
```

Saída

```
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)`

Mescla esse `DynamicFrame` com uma preparação `DynamicFrame` de acordo com as chaves primárias especificadas para identificar registros. Registros duplicados (com as mesmas chaves primárias) não são eliminados. Se não houver nenhum registro correspondente no quadro de preparação, todos os registros (incluindo os duplicados) serão retidos da origem. Se o quadro de preparação tiver registros correspondentes, os do quadro de preparação substituirão os da origem no AWS Glue.

- `stage_dynamic_frame`: o `DynamicFrame` de preparação para mesclar.
- `primary_keys` a lista de campos de chave primária para corresponder aos registros da fonte e quadros dinâmicos de preparação.
- `transformation_ctx`: uma string exclusiva usada para recuperar os metadados sobre a transformação atual (opcional).
- `options`: uma string de pares nome-valor JSON que fornecem informações adicionais para essa transformação. Esse argumento não é usado no momento.
- `info`: uma `String`. Qualquer string a ser associada a erros nessa transformação.
- `stageThreshold`: uma `Long`. O número de erros na transformação para a qual o processamento precisa apresentar falhas.
- `totalThreshold`: uma `Long`. O número total de erros até esta transformação (inclusive) para os quais o processamento precisa apresentar falhas.

Este método retorna um novo `DynamicFrame` obtido ao mesclar este `DynamicFrame` com a preparação `DynamicFrame`.

O `DynamicFrame` retornado contém registro A nestes casos:

- Se `A` existir no quadro de origem e no quadro de preparação, o `A` do quadro de preparação será retornado.
- Se `A` estiver na tabela de origem e `A.primaryKeys` não estiver no `stagingDynamicFrame`, `A` não será atualizado na tabela de preparação.

O quadro de origem e o quadro de preparação não precisam ter o mesmo esquema.

Exemplo: use mergeDynamicFrame para mesclar dois **`DynamicFrames`** com base em uma chave primária

O exemplo de código a seguir mostra como usar o método `mergeDynamicFrame` para mesclar um `DynamicFrame` com uma “preparação” `DynamicFrame` com base na chave primária `id`.

Exemplo de conjunto de dados

O exemplo usa dois `DynamicFrames` de um `DynamicFrameCollection` chamado `split_rows_collection`. Está é uma lista de limites no `split_rows_collection`.

```
dict_keys(['high', 'low'])
```

Código de exemplo

```
# 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()
```

Saída

```
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)`

Converte um `DynamicFrame` em um formulário que se encaixa em um banco de dados relacional. Relacionar um `DynamicFrame` é especialmente útil quando você deseja mover dados de um ambiente NoSQL como o DynamoDB para um banco de dados relacional como o MySQL.

A transformação gera uma lista de quadros separando colunas aninhadas e colunas de matriz dinâmica. A coluna de matriz dinâmica pode ser adicionada à tabela raiz usando a chave de união gerada durante a fase de desaninhamento.

- `root_table_name` – O nome da a tabela raiz.
- `staging_path`: o caminho onde o método pode armazenar partições de tabelas dinâmicas no formato CSV (opcional). As tabelas dinâmicas são lidas novamente nesse caminho.
- `options` – Um dicionário de parâmetros opcionais.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

Exemplo: usar relationalize para nivelar um esquema aninhado em um **`DynamicFrame`**

Este exemplo de código usa o método `relationalize` para nivelar um esquema aninhado em um formulário que se encaixa em um banco de dados relacional.

Exemplo de conjunto de dados

O exemplo usa um `DynamicFrame` chamado `legislators_combined` com o esquema a seguir. `legislators_combined` tem vários campos aninhados, como `links`, `images` e `contact_details` que serão nivelados pela transformação `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 exemplo

```
# 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()
```

Saída

A saída a seguir permite comparar o esquema do campo aninhado chamado `contact_details` com a tabela criada pela transformação `relationalize`. Observe que os registros da tabela apontam para a tabela principal usando uma chave estrangeira chamada `id` e uma coluna `index` que representa as posições da 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)`

Renomeia um campo neste `DynamicFrame` e retorna um novo `DynamicFrame` com o campo renomeado.

- `oldName` – O caminho completo para o nó que você quer renomear.

  Se o nome antigo contiver pontos, `RenameField` não funcionará a menos que você coloque acentos graves em torno dele (`). Por exemplo, para substituir `this.old.name` por `thisNewName`, você chamaria rename_field da seguinte maneira.

  ```
  newDyF = oldDyF.rename_field("`this.old.name`", "thisNewName")
  ```

- `newName` – O novo nome, como um caminho completo.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

Exemplo: usar rename_field para renomear campos em um **`DynamicFrame`**

Este exemplo de código usa o método `rename_field` para renomear campos em um `DynamicFrame`. Observe que o exemplo usa o encadeamento de métodos para renomear vários campos ao mesmo tempo.

> **Note**
>
> Para acessar o conjunto de dados usado neste exemplo, consulte Exemplo de código: juntar e relacionar dados e siga as instruções em Etapa 1: crawling de dados no bucket do Amazon S3.

Código de exemplo

```
# 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()
```

Saída

```
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)`

Resolve um tipo de escolha neste `DynamicFrame` e retorna o novo `DynamicFrame`.

- `specs`: uma lista de ambiguidades específicas para resolver, cada uma na forma de uma tupla: `(field_path, action)`.

  Há duas maneiras de usar `resolveChoice`. A primeira é usar o argumento `specs` para indicar uma sequência de colunas específicas e como resolvê-las. O outro modo para `resolveChoice` é usar o argumento `choice` para especificar uma única resolução para todos os `ChoiceTypes`.

  Valores para `specs` são especificados como tuplas compostas de pares `(field_path, action)`. O valor `field_path` identifica um elemento ambíguo específico, e o valor `action` identifica a resolução correspondente. A seguir estão as ações possíveis:

  - `cast:`*`type`*: tenta converter todos os valores para o tipo especificado. Por exemplo: `cast:int`.
  - `make_cols`: converte cada tipo distinto em uma coluna com o nome *`columnName_type`*. Ele resolve uma possível ambiguidade ao nivelar os dados. Por exemplo, se `columnA` puder ser `int` ou `string`, a resolução seria produzir duas colunas chamadas `columnA_int` e `columnA_string` no `DynamicFrame` resultante.
  - `make_struct`: resolve uma possível ambiguidade usando um `struct` para representar os dados. Por exemplo, se os dados em uma coluna pudessem ser um `int` ou `string`, usar a ação `make_struct` produziria uma coluna de estruturas no `DynamicFrame`. Cada estrutura contém um `int` e um `string`.
  - `project:`*`type`*: resolve uma possível ambiguidade projetando todos os dados para um dos possíveis tipos de dados. Por exemplo, se os dados em uma coluna pudessem ser um `int` ou `string`, usar a ação `project:string` produzirá uma coluna de estruturas no `DynamicFrame` resultante, onde todos os valores `int` foram convertidos em strings.

  Se o `field_path` identifica um array, insira colchetes vazios após o nome do array para evitar ambiguidades. Por exemplo, vamos supor que você esteja trabalhando com dados estruturados da seguinte maneira:

```
"myList": [
  { "price": 100.00 },
  { "price": "$100.00" }
]
```

Você pode selecionar a versão numérica em vez da versão string do preço definindo `field_path` como `"myList[].price"` e `action` como `"cast:double"`.

> **Note**
>
> É possível usar somente um dos parâmetros `specs` e `choice`. Se o parâmetro `specs` não for `None`, o parâmetro `choice` precisará ser uma string vazia. Por outro lado, se o `choice` não for uma string vazia, o parâmetro `specs` precisará ser `None`.

- `choice`: especifica uma única resolução para todos os `ChoiceTypes`. É possível usar essa ação em casos em que a lista completa de `ChoiceTypes` for desconhecida antes do runtime. Além das ações listadas anteriormente para `specs`, esse modo também aceita a seguinte ação:
  - `match_catalogChoiceType`: tenta converter cada para o tipo correspondente na tabela do Data Catalog especificada.
- `database`: banco de dados do Data Catalog a ser usado com a ação `match_catalog`.
- `table_name`: a tabela do Data Catalog a ser usada com a ação `match_catalog`.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados até esta transformação (inclusive) em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `catalog_id`: o ID do catálogo do Data Catalog sendo acessado (o ID da conta do Data Catalog). Quando definido como `None` (valor padrão), ele usa o ID do catálogo da conta de chamada.

Exemplo: usar resolveChoice para lidar com uma coluna que contém vários tipos

Este exemplo de código usa o método `resolveChoice` para especificar como lidar com uma coluna `DynamicFrame` que contém valores de vários tipos. O exemplo demonstra duas maneiras comuns de lidar com uma coluna com tipos diferentes:

- Converter a coluna em um único tipo de dados.
- Reter todos os tipos em colunas separadas.

Exemplo de conjunto de dados

> **Note**
>
> Para acessar o conjunto de dados usado neste exemplo, consulte Exemplo de código: preparo de dados usando ResolveChoice, Lambda e ApplyMapping e siga as instruções em Etapa 1: crawling de dados no bucket do Amazon S3.

O exemplo usa um `DynamicFrame` chamado `medicare` com o seguinte 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 exemplo

```
# 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()
```

Saída

```
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)`

Retorna um novo `DynamicFrame` que contém os campos selecionados.

- `paths`: uma lista de cadeias de caracteres. Cada cadeia de caracteres é um caminho para um nó de nível superior que você deseja selecionar.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

Exemplo: usar select_fields para criar um novo **`DynamicFrame`** com os campos escolhidos

O exemplo de código a seguir mostra como usar o método `select_fields` para criar um novo `DynamicFrame` com uma lista escolhida de campos de um `DynamicFrame` existente.

> **Note**
>
> Para acessar o conjunto de dados usado neste exemplo, consulte Exemplo de código: juntar e relacionar dados e siga as instruções em Etapa 1: crawling de dados no bucket do 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()
```

Saída

```
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 colunas aninhadas em um `DynamicFrame` que estão especificamente na estrutura JSON do DynamoDB e retorna um novo `DynamicFrame` simplificado. Se houver vários tipos ou um tipo de mapa em um tipo de lista, os elementos na lista não serão simplificados. Observe que esse é um tipo específico de transformação que se comporta de forma diferente da transformação `unnest` comum e requer que os dados já estejam na estrutura JSON do DynamoDB. Para mais informações, consulte JSON do DynamoDB.

Por exemplo, o esquema de uma leitura de uma exportação com a estrutura JSON do DynamoDB pode se parecer com o seguinte:

```
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
```

A transformação `simplify_ddb_json()` converteria isso em:

```
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
```

Exemplo: use simplify_ddb_json para invocar uma simplificação JSON do DynamoDB

Esse exemplo de código usa o método `simplify_ddb_json` para utilizar o conector de exportação para DynamoDB do AWS Glue, invocar uma simplificação JSON do DynamoDB e imprimir o número de partições.

Código de exemplo

```
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={})`

Grava registros de exemplo em um destino específico para ajudar você a verificar as transformações realizadas pelo seu trabalho.

- `path`: o caminho para o destino no qual a gravação será feita (obrigatório).
- `options`: pares de chave-valor que especificam opções (opcional). A opção `"topk"` especifica que os primeiros registros k devem ser gravados. A opção `"prob"` especifica a probabilidade (como um número decimal) de escolher um determinado registro. Ele pode ser usado na seleção de registros para gravar.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).

Exemplo: usar spigot para gravar campos de exemplo de um **`DynamicFrame`** no Amazon S3

Este exemplo de código usa o método `spigot` para gravar registros de amostra em um bucket do Amazon S3 depois de aplicar a transformação `select_fields`.

Exemplo de conjunto de dados

> **Note**
>
> Para acessar o conjunto de dados usado neste exemplo, consulte Exemplo de código: juntar e relacionar dados e siga as instruções em Etapa 1: crawling de dados no bucket do Amazon S3.

O exemplo usa um `DynamicFrame` chamado `persons` com o seguinte 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 exemplo

```
# 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}
)
```

Saída

Veja a seguir um exemplo dos dados gravados por `spigot` no Amazon S3. Como o código de exemplo especificou `options={"topk": 10}`, os dados de exemplo contêm os primeiros 10 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)`

Retorna um novo `DynamicFrameCollection` que contém dois `DynamicFrames`. O primeiro `DynamicFrame` contém todos os nós que foram separados, e o segundo contém os nós restantes.

- `paths` – Uma lista de strings, cada uma é um caminho completo para um nó que você quer separar em um novo `DynamicFrame`.
- `name1` – Uma string de nome para `DynamicFrame` a ser separado.
- `name2` – Uma string de nome para `DynamicFrame` que permanece após os nós especificados terem sido separados.

- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

Exemplo: usar split_fields para dividir os campos selecionados em um **`DynamicFrame`** separado

Este exemplo de código usa o método `split_fields` para dividir uma lista de campos especificados em uma lista separada `DynamicFrame`.

Exemplo de conjunto de dados

O exemplo usa um `DynamicFrame` chamado `l_root_contact_details` que é de uma coleção chamada `legislators_relationalized`.

`l_root_contact_details` tem o seguinte esquema e 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 exemplo

```
# 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()
```

Saída

```
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)`

Separa uma ou mais linhas em um `DynamicFrame` em um novo `DynamicFrame`.

O método retorna um novo `DynamicFrameCollection` que contém dois `DynamicFrames`. O primeiro `DynamicFrame` contém todas as linhas que foram separadas, e o segundo contém as linhas restantes.

- `comparison_dict`: um dicionário em que a chave é o caminho para uma coluna e o valor é outro dicionário para mapear comparadores a valores aos quais os valores das colunas são comparados. Por exemplo, `{"age": {">": 10, "<": 20}}` separa todas as linhas cujo valor na coluna de idade é maior do que 10 e menor do que 20.
- `name1` – Uma string de nome para `DynamicFrame` a ser separado.
- `name2` – Uma string de nome para `DynamicFrame` que permanece após os nós especificados terem sido separados.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

Exemplo: usar split_rows para dividir linhas em um **`DynamicFrame`**

Este exemplo de código usa o método `split_rows` para dividir linhas em um `DynamicFrame` com base no valor do campo `id`.

Exemplo de conjunto de dados

O exemplo usa um `DynamicFrame` chamado `l_root_contact_details` que é selecionado em uma coleção chamada `legislators_relationalized`.

`l_root_contact_details` tem o seguinte esquema e 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 exemplo

```
# 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()
```

Saída

```
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)`

Descompacta (reformata) um campo de string em um `DynamicFrame` e retorna um novo `DynamicFrame` que contém os `DynamicRecords` descompactados.

Um `DynamicRecord` representa um registro lógico em um `DynamicFrame`. É semelhante a uma linha em um `DataFrame` do Apache Spark, exceto pelo fato de que pode se autodescrever e ser usado para dados que não estejam em conformidade com um esquema fixo.

- `path` – Um caminho completo para o nó de string que você quer descompactar.
- `format`: uma especificação de formato (opcional). Usado para uma conexão do Amazon S3 ou do AWS Glue com suporte a vários formatos. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para conhecer os formatos compatíveis.

- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `options`: um ou mais dos seguintes itens:
  - `separator`: uma string que contém o caractere de separação.
  - `escaper`: uma string que contém o caractere de escape.
  - `skipFirst`: um valor booleano que indica se a primeira instância deve ser ignorada.
  - `withSchema`: uma string contendo uma representação JSON do esquema do nó. O formato da representação JSON de um esquema é definido pela saída de `StructType.json()`.
  - `withHeader`: um valor booleano que indica se há um cabeçalho incluído.

Exemplo: usar unbox para descompactar um campo de string em um struct

Este exemplo de código usa o método `unbox` para desempacotar ou reformatar um campo de string em um `DynamicFrame` em um campo do tipo struct.

Exemplo de conjunto de dados

O exemplo usa um `DynamicFrame` chamado `mapped_with_string` com os seguintes esquema e entradas:

Observe o campo chamado `AddressString`. Esse é o campo que o exemplo descompacta em um struct.

```
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 exemplo

```
# 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()
```

Saída

```
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
```

união

`union(frame1, frame2, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)`

União de dois DynamicFrames. Retorna DynamicFrame contendo todos os registros dos dois DynamicFrames de entrada. Essa transformação pode retornar resultados diferentes da união de dois DataFrames com dados equivalentes. Se você precisar do comportamento de união do Spark DataFrame, considere usar `toDF`.

- `frame1` - Primeiro DynamicFrame para a união.
- `frame2` - Segundo DynamicFrame para a união.
- `transformation_ctx` - (opcional) Uma string exclusiva usada para identificar estatísticas/ informações de estado
- `info` - (opcional) Qualquer string a ser associada a erros na transformação
- `stageThreshold` - (opcional) Número máximo de erros na transformação até que o processamento ocorra um erro

- `totalThreshold` - (opcional) Número máximo de erros totais até que o processamento apresente erros.

unnest

`unnest(transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Desfaz o aninhamento de objetos em um `DynamicFrame`, transformando-os em objetos de nível superior, e retorna um novo `DynamicFrame` não aninhado.

- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold`: o número de erros encontrados durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.
- `totalThreshold`: o número de erros encontrados antes e durante essa transformação em que o processo deve falhar (opcional). O padrão é zero, o que indica que o processo não deve falhar.

Exemplo: usar unnest para transformar campos aninhados em campos de nível superior

Este exemplo de código usa o método `unnest` para nivelar todos os campos aninhados em um `DynamicFrame` em campos de nível superior.

Exemplo de conjunto de dados

O exemplo usa um `DynamicFrame` chamado `mapped_medicare` com o esquema a seguir. Observe que o campo `Address` é o único campo que contém dados aninhados.

```
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 exemplo

```
# 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()
```

Saída

```
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

Desaninha colunas aninhadas em um `DynamicFrame` que estão especificamente na estrutura JSON do DynamoDB e retorna um novo `DynamicFrame` não aninhado. Colunas que pertençam a uma matriz de tipos de estrutura não serão desaninhadas. Observe que esse é um tipo específico de transformação de desaninhamento que se comporta diferentemente da transformação `unnest` comum e requer que os dados já estejam na estrutura JSON do DynamoDB. Para mais informações, consulte JSON do DynamoDB.

`unnest_ddb_json(transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string a ser associada com o erro na geração de relatórios desta transformação (opcional).
- `stageThreshold` – O número de erros encontrados durante esta transformação em que o processo deve falhar (opcional: zero por padrão, indicando que o processo não deve apresentar falha).
- `totalThreshold` – O número de erros encontrados incluindo esta transformação em que o processo deve falhar (opcional: zero por padrão, indicando que o processo não deve apresentar falha).

Por exemplo, o esquema de uma leitura de uma exportação com a estrutura JSON do DynamoDB pode ter a seguinte aparência:

```
root
|-- Item: struct
|    |-- ColA: struct
|    |    |-- S: string
|    |-- ColB: struct
|    |    |-- S: string
|    |-- ColC: struct
|    |    |-- N: string
|    |-- ColD: struct
|    |    |-- L: array
|    |    |    |-- element: null
```

A transformação `unnest_ddb_json()` converteria isso em:

```
root
|-- ColA: string
|-- ColB: string
|-- ColC: string
|-- ColD: array
|    |-- element: null
```

O exemplo de código a seguir mostra como usar o conector de exportação para DynamoDB do AWS Glue, invocar um desaninhamento de JSON do DynamoDB e imprimir o número de partições:

```
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)`

Obtém um DataSink(object) do tipo de conexão especificado em GlueContext classe deste `DynamicFrame`, e o usa para formatar e gravar o conteúdo desse `DynamicFrame`. Retorna o novo `DynamicFrame` formatado e gravado conforme especificado.

- `connection_type`: o tipo de conexão a ser usado. Os valores válidos incluem `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` e `oracle`.
- `connection_options`: a opção de conexão a ser usada (opcional). Para um `connection_type` do `s3`, um caminho do Amazon S3 é definido.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

Para conexões JDBC, várias propriedades devem ser definidas. Observe que o nome do banco de dados deve fazer parte do URL. Ele também pode ser incluído nas opções de conexão.

> **Warning**
>
> Não é recomendável armazenar senhas no script. Considere usar `boto3` para recuperá-los do AWS Secrets Manager ou do catálogo de dados do AWS Glue.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

- `format`: uma especificação de formato (opcional). Essa ação é usada para um Amazon Simple Storage Service (Amazon S3) ou uma conexão do AWS Glue que ofereça suporte a vários formatos. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `format_options`: as opções de formato para o formato especificado. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `accumulator_size`: o tamanho cumulativo a ser usado, em bytes (opcional).

— erros —

- assertErrorThreshold
- errorsAsDynamicFrame
- errorsCount
- stageErrorsCount

assertErrorThreshold

`assertErrorThreshold( )`: uma afirmação para erros nas transformações que criaram este `DynamicFrame`. Retorna um `Exception` do `DataFrame` subjacente.

errorsAsDynamicFrame

`errorsAsDynamicFrame( )` – Retorna um `DynamicFrame` com registros de erro aninhados.

Exemplo: usar errorsAsDynamicFrame para visualizar registros de erros

O código de exemplo a seguir mostra como usar o método `errorsAsDynamicFrame` para visualizar um registro de erro para um `DynamicFrame`.

Exemplo de conjunto de dados

O exemplo usa o conjunto de dados a seguir, que você pode carregar para o Amazon S3 como JSON. O segundo registro está mal formado. Dados malformados geralmente interrompem a análise de arquivos quando usamos o SparkSQL. No entanto, o `DynamicFrame` reconhece problemas de malformação e transforma linhas malformadas em registros de erros que você pode solucionar individualmente.

```
{"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 exemplo

```
# 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]))
```

Saída

```
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( )` – Retorna o número total de erros em um `DynamicFrame`.

stageErrorsCount

`stageErrorsCount` – Retorna o número de erros que ocorreram no processo de criação deste `DynamicFrame`.

Classe DynamicFrameCollection

Um `DynamicFrameCollection` é um dicionário de objetos Classe DynamicFrame, em que as chaves são os nomes de `DynamicFrames` e os valores são os objetos `DynamicFrame`.

__init__

`__init__(dynamic_frames, glue_ctx)`

- `dynamic_frames` – Um dicionários de objetos Classe DynamicFrame.
- `glue_ctx` – Um objeto GlueContext classe.

Chaves

`keys( )` – Retorna uma lista das chaves nesta coleção, que geralmente consiste nos nomes de valores correspondentes `DynamicFrame`.

Valores

`values(key)` – Retorna uma lista de valores `DynamicFrame` nesta coleção.

Selecionar

`select(key)`

Retorna o `DynamicFrame` que corresponde à chave especificada (que geralmente é o nome do `DynamicFrame`).

- `key` – Uma chave no `DynamicFrameCollection`, que geralmente representa o nome de um `DynamicFrame`.

Mapa

`map(callable, transformation_ctx="")`

Usa uma função transmitida para criar e retornar um novo `DynamicFrameCollection` com base em `DynamicFrames` nesta coleção.

- `callable` – Uma função que usa `DynamicFrame` e o contexto de transformação especificado como parâmetros e retorna um `DynamicFrame`.
- `transformation_ctx` – Um contexto de transformação a ser usado por algo que pode ser chamado (opcional).

Flatmap

`flatmap(f, transformation_ctx="")`

Usa uma função transmitida para criar e retornar um novo `DynamicFrameCollection` com base em `DynamicFrames` nesta coleção.

- `f` – Uma função que usa um `DynamicFrame` como um parâmetro e retorna um `DynamicFrame` ou `DynamicFrameCollection`.
- `transformation_ctx` – Um contexto de transformação a ser usado pela função (opcional).

Classe DynamicFrameWriter

Métodos

- __init__
- from_options
- from_catalog
- from_jdbc_conf

__init__

`__init__(glue_context)`

- `glue_context` – O GlueContext classe a ser usado.

from_options

`from_options(frame, connection_type, connection_options={}, format=None, format_options={}, transformation_ctx="")`

Escreve um `DynamicFrame` usando a conexão e o formato especificados.

- `frame` – O `DynamicFrame` a ser escrito.
- `connection_type`: o tipo de conexão. Os valores válidos incluem `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` e `oracle`.

- `connection_options` – Opções de conexão, como caminho e tabela de banco de dados (opcional). Para um `connection_type` do `s3`, um caminho do Amazon S3 é definido.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

Para conexões JDBC, várias propriedades devem ser definidas. Observe que o nome do banco de dados deve fazer parte do URL. Ele também pode ser incluído nas opções de conexão.

> **Warning**
>
> Não é recomendável armazenar senhas no script. Considere usar `boto3` para recuperá-los do AWS Secrets Manager ou do catálogo de dados do AWS Glue.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

A propriedade `dbtable` é o nome da tabela JDBC. Para armazenamentos de dados JDBC que oferecem suporte a esquemas dentro de um banco de dados, especifique `schema.table-name`. Se um esquema não for fornecido, o esquema "público" padrão será usado.

Para ter mais informações, consulte Tipos e opções de conexão para ETL no AWS Glue para Spark.

- `format`: uma especificação de formato (opcional). Essa ação é usada para um Amazon Simple Storage Service (Amazon S3) ou uma conexão do AWS Glue que ofereça suporte a vários formatos. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `format_options`: as opções de formato para o formato especificado. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `transformation_ctx` – Um contexto de transformação a ser usado (opcional).

from_catalog

`from_catalog(frame, name_space, table_name, redshift_tmp_dir="", transformation_ctx="")`

Escreve um `DynamicFrame` usando o banco de dados do catálogo e o nome da tabela especificados.

- `frame` – O `DynamicFrame` a ser escrito.
- `name_space` – O mecanismo de banco de dados a ser usado.
- `table_name` – O `table_name` a ser usado.
- `redshift_tmp_dir`: um diretório temporário do Amazon RedShift a ser usado (opcional).
- `transformation_ctx` – Um contexto de transformação a ser usado (opcional).
- `additional_options`: opções adicionais fornecidas ao AWS Glue.

  Para gravar em tabelas governadas pelo Lake Formation, você pode usar estas opções adicionais:

  - `transactionId`: (string) o ID da transação no qual fazer a gravação na tabela governada. Esta transação já não pode ser confirmada ou anulada, ou a gravação falhará.
  - `callDeleteObjectsOnCancel` : (booleano, opcional) Se definido como `true` (padrão), o AWS Glue chama automaticamente a API `DeleteObjectsOnCancel` após o objeto ser gravado no Amazon S3. Para obter mais informações, consulte DeleteObjectsOnCancel no Guia do desenvolvedor do AWS Lake Formation.

  Example Exemplo: gravação em uma tabela governada no 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="")`

Escreve um `DynamicFrame` usando as informações de conexão JDBC especificadas.

- `frame` – O `DynamicFrame` a ser escrito.
- `catalog_connection` – Uma conexão de catálogo a ser usada.
- `connection_options` – Opções de conexão, como caminho e tabela de banco de dados (opcional).
- `redshift_tmp_dir`: um diretório temporário do Amazon RedShift a ser usado (opcional).
- `transformation_ctx` – Um contexto de transformação a ser usado (opcional).

Exemplo para write_dynamic_frame

Este exemplo grava a saída localmente usando um `connection_type` do S3 com um argumento de caminho POSIX em `connection_options`, o que permite gravar no armazenamento local.

```
glueContext.write_dynamic_frame.from_options(\
frame = dyf_splitFields,\
connection_options = {'path': '/home/glue/GlueLocalOutput/'},\
connection_type = 's3',\
format = 'json')
```

DynamicFrameReader classe

— métodos —

- __init__
- from_rdd
- from_options
- from_catalog

__init__

`__init__(glue_context)`

- `glue_context` – O GlueContext classe a ser usado.

from_rdd

`from_rdd(data, name, schema=None, sampleRatio=None)`

Lê um `DynamicFrame` de um Conjunto de dados resiliente distribuído (RDD).

- `data` – O conjunto de dados onde a leitura será feita.
- `name` – O nome onde a leitura será feita.
- `schema` – O esquema a ser lido (opcional).
- `sampleRatio` – A proporção da amostra (opcional).

from_options

`from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx="")`

Lê um `DynamicFrame` usando a conexão e o formato especificados.

- `connection_type`: o tipo de conexão. Os valores válidos incluem `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle`, `dynamodb` e `snowflake`.
- `connection_options` – Opções de conexão, como caminho e tabela de banco de dados (opcional). Para obter mais informações, consulte Tipos e opções de conexão para ETL no AWS Glue for Spark. Para um `connection_type` do `s3`, os caminhos do Amazon S3 são definidos em uma matriz.

  ```
  connection_options = {"paths": [ "s3://mybucket/object_a", "s3://mybucket/object_b"]}
  ```

  Para conexões JDBC, várias propriedades devem ser definidas. Observe que o nome do banco de dados deve fazer parte do URL. Ele também pode ser incluído nas opções de conexão.

  > **Warning**
  >
  > Não é recomendável armazenar senhas no script. Considere usar `boto3` para recuperá-los do AWS Secrets Manager ou do AWS Glue Data Catalog.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-
path"}
```

Para uma conexão JDBC que realiza leituras paralelas, você pode definir a opção hashfield. Por exemplo:

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-
path" , "hashfield": "month"}
```

Para ter mais informações, consulte Leitura de tabelas JDBC em paralelo.

- `format`: uma especificação de formato (opcional). Essa ação é usada para um Amazon Simple Storage Service (Amazon S3) ou uma conexão do AWS Glue que ofereça suporte a vários formatos. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `format_options`: as opções de formato para o formato especificado. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `push_down_predicate`: filtra partições sem a necessidade de listar e ler todos os arquivos no seu conjunto de dados. Para obter mais informações, consulte Pré-filtragem usando a aplicação de predicados.

from_catalog

`from_catalog(database, table_name, redshift_tmp_dir="", transformation_ctx="", push_down_predicate="", additional_options={})`

Lê um `DynamicFrame` usando o namespace do catálogo e o nome da tabela especificados.

- `database` – O banco de dados onde a leitura será feita.
- `table_name` – O nome da tabela onde a leitura será feita.
- `redshift_tmp_dir`: um diretório temporário do Amazon Redshift a ser usado (opcional se não estiver lendo dados do Redshift).

- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `push_down_predicate`: filtra partições sem a necessidade de listar e ler todos os arquivos no seu conjunto de dados. Para ter mais informações, consulte Pré-filtragem usando a aplicação de predicados.
- `additional_options`: opções adicionais fornecidas ao AWS Glue.
  - Para usar uma conexão JDBC que realiza leituras paralelas, você pode definir as opções `hashfield`, `hashexpression` ou `hashpartitions`. Por exemplo:

    ```
    additional_options = {"hashfield": "month"}
    ```

    Para ter mais informações, consulte Leitura de tabelas JDBC em paralelo.
  - Para transmitir uma expressão de catálogo para filtrar com base nas colunas de índice, você pode consultar o opção `catalogPartitionPredicate`.

    `catalogPartitionPredicate`: você pode transmitir uma expressão de catálogo para filtrar com base nas colunas de índice. Isso leva a filtragem para o lado do servidor. Para obter mais informações, consulte Índices de partição do AWS Glue. Observe que `push_down_predicate` e `catalogPartitionPredicate` usam sintaxes diferentes. O primeiro usa a sintaxe padrão do Spark SQL e o outro usa o analisador JSQL.

    Para ter mais informações, consulte Gerenciar partições para saída de ETL no AWS Glue.

GlueContext classe

Envolve o SparkContextobjeto Apache Spark e, assim, fornece mecanismos para interagir com a plataforma Apache Spark.

__init__

`__init__(sparkContext)`

- `sparkContext` – O contexto do Apache Spark a ser usado.

Criando

- __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)`

Cria um objeto `DataSource` que pode ser usado para ler `DynamicFrames` a partir de fontes externas.

- `connection_type`: o tipo de conexão a ser usado, por exemplo, Amazon Simple Storage Service (Amazon S3), Amazon Redshift e JDBC. Os valores válidos incluem `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle` e `dynamodb`.
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `options` – Uma coleção de pares nome-valor opcionais. Para ter mais informações, consulte Tipos e opções de conexão para ETL no AWS Glue para Spark.

Veja a seguir um exemplo de uso do `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="")`

Retorna um `DynamicFrame` que é criado a partir de um conjunto de dados resiliente distribuído (RDD) do Apache Spark.

- `data` – A fonte de dados a ser usada.
- `name` – O nome dos dados a serem usados.
- `schema` – O esquema a ser usado (opcional).
- `sample_ratio` – A proporção da amostra a ser usada (opcional).
- `transformation_ctx` – O contexto de transformação a ser usado (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)`

Retorna um `DynamicFrame` que é criado usando um banco de dados do Data Catalog e o nome da tabela. Ao usar esse método, você fornece `format_options` por meio das propriedades da tabela especificada do AWS Glue Data Catalog e outras opções por meio do `additional_options` argumento.

- `Database` – O banco de dados onde a leitura será feita.
- `table_name` – O nome da tabela onde a leitura será feita.
- `redshift_tmp_dir`: um diretório temporário do Amazon Redshift a ser usado (opcional).
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `push_down_predicate`: filtra partições sem a necessidade de listar e ler todos os arquivos no seu conjunto de dados. Para fontes e limitações suportadas, consulte Otimizando leituras com push down no AWS Glue ETL. Para ter mais informações, consulte Pré-filtragem usando a aplicação de predicados.
- `additional_options` – Uma coleção de pares nome-valor opcionais. As opções possíveis incluem as listadas em Tipos e opções de conexão para ETL no AWS Glue para Spark, exceto para `endpointUrl`, `streamName`, `bootstrap.servers`, `security.protocol`, `topicName`, `classification` e `delimiter`. Outra opção suportada é `catalogPartitionPredicate`:

  `catalogPartitionPredicate`: você pode transmitir uma expressão de catálogo para filtrar com base nas colunas de índice. Isso leva a filtragem para o lado do servidor. Para obter mais informações, consulte Índices de partição do AWS Glue. Observe que `push_down_predicate` e `catalogPartitionPredicate` usam sintaxes diferentes. O primeiro usa a sintaxe padrão do Spark SQL e o outro usa o analisador JSQL.

- `catalog_id`: o ID do catálogo (ID da conta) do Data Catalog que está sendo acessado. Quando None (Nenhum), o ID da conta do chamador padrão é usado.

create_dynamic_frame_from_options

`create_dynamic_frame_from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx = "")`

Retorna um `DynamicFrame` criado com a conexão e o formato especificados.

- `connection_type`: o tipo de conexão, por exemplo, Amazon S3, Amazon Redshift e JDBC. Os valores válidos incluem `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle` e `dynamodb`.
- `connection_options`: opções de conexão, como caminhos e tabela de banco de dados (opcional). Para um `connection_type` do `s3`, uma lista de caminhos do Amazon S3 é definida.

```
connection_options = {"paths": ["s3://aws-glue-target/temp"]}
```

Para conexões JDBC, várias propriedades devem ser definidas. Observe que o nome do banco de dados deve fazer parte do URL. Ele também pode ser incluído nas opções de conexão.

> **Warning**
>
> Não é recomendável armazenar senhas no script. Considere usar `boto3` para recuperá-los do AWS Secrets Manager ou do AWS Glue Data Catalog.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

A propriedade `dbtable` é o nome da tabela JDBC. Para armazenamentos de dados JDBC que oferecem suporte a esquemas dentro de um banco de dados, especifique `schema.table-name`. Se um esquema não for fornecido, o esquema "público" padrão será usado.

Para ter mais informações, consulte Tipos e opções de conexão para ETL no AWS Glue para Spark.

- `format`— Uma especificação de formato. É usado para uma conexão do Amazon S3 ou do AWS Glue com suporte para vários formatos. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `format_options`: as opções de formato para o formato especificado. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `push_down_predicate`: filtra partições sem a necessidade de listar e ler todos os arquivos no seu conjunto de dados. Para fontes e limitações suportadas, consulte Otimizando leituras com push down no AWS Glue ETL. Para obter mais informações, consulte Pré-filtragem usando a aplicação de predicados.

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)`

Retorna um modelo `DynamicFrame` que é criado usando um banco de dados do Data Catalog e o nome da tabela. O `DynamicFrame` contém apenas os primeiros `num` registros de uma fonte de dados.

- `database` – O banco de dados onde a leitura será feita.
- `table_name` – O nome da tabela onde a leitura será feita.
- `num` - O número máximo de registros no quadro dinâmico de amostra retornado.
- `redshift_tmp_dir`: um diretório temporário do Amazon Redshift a ser usado (opcional).
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `push_down_predicate`: filtra partições sem a necessidade de listar e ler todos os arquivos no seu conjunto de dados. Para ter mais informações, consulte Pré-filtragem usando a aplicação de predicados.
- `additional_options` – Uma coleção de pares nome-valor opcionais. As opções possíveis incluem as listadas em Tipos e opções de conexão para ETL no AWS Glue para Spark, exceto para `endpointUrl`, `streamName`, `bootstrap.servers`, `security.protocol`, `topicName`, `classification` e `delimiter`.

- `sample_options` - Parâmetros para controlar o comportamento de amostragem (opcional). Parâmetros disponíveis atuais para fontes do Amazon S3:
  - `maxSamplePartitions` - O número máximo de partições que a amostragem lerá. O valor padrão é 10
  - `maxSampleFilesPerPartition` - O número máximo de partições que a amostragem lerá em uma partição. O valor padrão é 10

    Esses parâmetros ajudam a reduzir o tempo consumido pela listagem de arquivos. Por exemplo, suponha que o conjunto de dados tenha 1.000 partições e cada partição tenha 10 arquivos. Se você definir `maxSamplePartitions`= 10 e `maxSampleFilesPerPartition`= 10, em vez de listar todos os 10.000 arquivos, a amostragem listará e lerá apenas as primeiras 10 partições com os primeiros 10 arquivos em cada: 10*10 = 100 arquivos no total.
- `catalog_id`: o ID do catálogo do Data Catalog sendo acessado (o ID da conta do Data Catalog). Definido como `None` por padrão. O `None` é definido como padrão para o ID do catálogo da conta de chamada no serviço.

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 = "")`

Retorna um modelo `DynamicFrame` criado com a conexão e o formato especificados. O `DynamicFrame` contém apenas os primeiros `num` registros de uma fonte de dados.

- `connection_type`: o tipo de conexão, por exemplo, Amazon S3, Amazon Redshift e JDBC. Os valores válidos incluem `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle` e `dynamodb`.
- `connection_options`: opções de conexão, como caminhos e tabela de banco de dados (opcional). Para ter mais informações, consulte Tipos e opções de conexão para ETL no AWS Glue para Spark.
- `num` - O número máximo de registros no quadro dinâmico de amostra retornado.
- `sample_options` - Parâmetros para controlar o comportamento de amostragem (opcional). Parâmetros disponíveis atuais para fontes do Amazon S3:
  - `maxSamplePartitions` - O número máximo de partições que a amostragem lerá. O valor padrão é 10

- `maxSampleFilesPerPartition` - O número máximo de partições que a amostragem lerá em uma partição. O valor padrão é 10

  Esses parâmetros ajudam a reduzir o tempo consumido pela listagem de arquivos. Por exemplo, suponha que o conjunto de dados tenha 1.000 partições e cada partição tenha 10 arquivos. Se você definir `maxSamplePartitions`= 10 e `maxSampleFilesPerPartition`= 10, em vez de listar todos os 10.000 arquivos, a amostragem listará e lerá apenas as primeiras 10 partições com os primeiros 10 arquivos em cada: 10*10 = 100 arquivos no total.

- `format`— Uma especificação de formato. É usado para uma conexão do Amazon S3 ou do AWS Glue com suporte para vários formatos. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `format_options`: as opções de formato para o formato especificado. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `push_down_predicate`: filtra partições sem a necessidade de listar e ler todos os arquivos no seu conjunto de dados. Para ter mais informações, consulte Pré-filtragem usando a aplicação de predicados.

add_ingestion_time_columns

`add_ingestion_time_columns(dataFrame, timeGranularity = "")`

Acrescenta colunas de tempo de ingestão, como `ingest_year`, `ingest_month`, `ingest_day`, `ingest_hour`, `ingest_minute`, para o `DataFrame` de entrada. Essa função é gerada automaticamente no script gerado pelo AWS Glue, quando você especifica uma tabela do catálogo de dados com o Amazon S3 como destino. Essa função atualiza automaticamente a partição com colunas de tempo de ingestão na tabela de saída. Isso permite que os dados de saída sejam particionados automaticamente no tempo de ingestão sem exigir colunas de tempo de ingestão explícitas nos dados de entrada.

- `dataFrame`: o `dataFrame` ao qual anexar as colunas de tempo de ingestão.
- `timeGranularity`: o detalhamento das colunas de tempo. Os valores válidos são “`day`”, “`hour`” e “`minute`”. Por exemplo, se “`hour`” é transmitido para a função, o `dataFrame` original terá as colunas de tempo “`ingest_year`”, “`ingest_month`”, “`ingest_day`” e “`ingest_hour`” anexadas.

Retorna o quadro de dados após anexar as colunas de detealhamento de tempo.

Exemplo:

```
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 = {})`

Retorna um `DataFrame` que é criado usando informações de uma tabela do Data Catalog.

- `database`: o banco de dados do Data Catalog do qual fazer a leitura.
- `table_name`: o nome da tabela do Data Catalog da qual fazer a leitura.
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `additional_options` – Uma coleção de pares nome-valor opcionais. As opções possíveis incluem as listadas em Tipos e opções de conexão para ETL no AWS Glue para Spark para fontes de transmissão, como `startingPosition`, `maxFetchTimeInMs` e `startingOffsets`.
  - `useSparkDataSource`— Quando definido como true, força o AWS Glue a usar a API nativa do Spark Data Source para ler a tabela. A API do Spark Data Source é compatível nos seguintes formatos: AVRO, binário, CSV, JSON, ORC, Parquet e texto. Em uma tabela do Data Catalog, você especifica o formato usando a propriedade `classification`. Para saber mais sobre a API do Spark Data Source, consulte a documentação oficial do Apache Spark.

    O uso de `create_data_frame_from_catalog` com `useSparkDataSource` apresenta os seguintes benefícios:

    - Retorna diretamente um `DataFrame` e fornece uma alternativa a `create_dynamic_frame.from_catalog().toDF()`.
    - Oferece suporte ao controle AWS Lake Formation de permissões em nível de tabela para formatos nativos.
    - Suporta a leitura de formatos de data lake sem AWS Lake Formation controle de permissão em nível de tabela. Para ter mais informações, consulte Usar estruturas de data lake com trabalhos do AWS Glue ETL.

Ao ativar`useSparkDataSource`, você também pode adicionar qualquer uma das opções de fonte de dados do Spark `additional_options` conforme necessário. AWS O Glue passa essas opções diretamente para o leitor Spark.

- `useCatalogSchema`— Quando definido como true, o AWS Glue aplica o esquema do Catálogo de Dados ao resultado`DataFrame`. Caso contrário, o leitor infere o esquema a partir dos dados. Quando você habilita o `useCatalogSchema`, deve também definir `useSparkDataSource` como true.

Limitações

Considere as seguintes limitações ao usar a opção `useSparkDataSource`:

- Quando você usa`useSparkDataSource`, o AWS Glue cria um novo `DataFrame` em uma sessão separada do Spark que é diferente da sessão original do Spark.
- A filtragem de DataFrame partições do Spark não funciona com os seguintes recursos do AWS Glue.
  - Marcadores de trabalho
  - Excluir classes de armazenamento do Amazon S3
  - Predicados de partição de catálogo

  Para usar a filtragem de partições com esses recursos, você pode usar o predicado pushdown AWS Glue. Para ter mais informações, consulte Pré-filtragem usando a aplicação de predicados. A filtragem em colunas não particionadas não é afetada.

  O script de exemplo a seguir demonstra a maneira incorreta de realizar a filtragem de partições com a opção `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'")
```

O exemplo de script a seguir demonstra a maneira correta de usar um predicado de pushdown para realizar a filtragem de partições com a opção `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'")
```

Exemplo: criar uma tabela CSV usando o leitor de fonte de dados do 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 = "")`

Essa API foi descontinuada. No lugar dela, use a API `getSource()`. Retorna um `DataFrame` criado com a conexão e o formato especificados. Use essa função apenas com fontes de transmissão do AWS Glue.

- `connection_type`: o tipo de conexão de transmissão. Os valores válidos são `kinesis` e `kafka`.
- `connection_options`: opções de conexão, que são diferentes para Kinesis e Kafka. Você pode encontrar a lista de todas as opções de conexão para cada origem dos dados de transmissão em Tipos e opções de conexão para ETL no AWS Glue para Spark. Observe as seguintes diferenças nas opções de conexão de transmissão:
  - As fontes de transmissão do Kinesis exigem `streamARN`, `startingPosition`, `inferSchema` e `classification`.
  - As fontes de transmissão do Kafka exigem `connectionName`, `topicName`, `startingOffsets`, `inferSchema` e `classification`.
- `format`— Uma especificação de formato. É usado para uma conexão do Amazon S3 ou do AWS Glue com suporte para vários formatos. Para obter informações sobre os formatos compatíveis, consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark.
- `format_options`: as opções de formato para o formato especificado. Para obter informações sobre as opções de formato compatíveis, consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark.
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).

Exemplo para fonte de transmissão do 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)
```

Exemplo para fonte de transmissão do 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)`

Aplica a `batch_function` transmitida para cada microlote lido a partir da fonte de transmissão.

- `frame`— O DataFrame que contém o microlote atual.
- `batch_function`: uma função que será aplicada para cada microlote.
- `options`: uma coleção de pares de chave-valor que contém informações sobre como processar microlotes. São necessárias as seguintes opções:
  - `windowSize`: a quantidade de tempo gasto no processamento de cada lote.
  - `checkpointLocation`: o local onde os pontos de verificação são armazenados para o trabalho de ETL de transmissão.
  - `batchMaxRetries`: o número máximo de novas tentativas deste lote em caso de falha. O valor padrão é 3. Essa opção só pode ser configurada para o Glue versão 2.0 e posterior.

Exemplo:

```
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
        )
```

Trabalhar com conjuntos de dados no 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="")`

Exclui arquivos do Amazon S3 para as tabelas e os bancos de dados do catálogo especificados. Se todos os arquivos em uma partição forem excluídos, essa partição também será removida do catálogo.

Se quiser recuperar objetos excluídos, você pode habilitar o versionamento de objeto no bucket do Amazon S3. Quando um objeto é excluído de um bucket que não tem o versionamento de objeto habilitado, o objeto não pode ser recuperado. Para obter mais informações sobre como recuperar objetos excluídos em um bucket habilitado para versionamento, consulte Como posso recuperar um objeto do Amazon S3 que foi excluído? na Central de Conhecimento do AWS Support .

- `catalog_id`: o ID do catálogo do Data Catalog sendo acessado (o ID da conta do Data Catalog). Definido como `None` por padrão. O `None` é definido como padrão para o ID do catálogo da conta de chamada no serviço.
- `database` – O mecanismo de banco de dados a ser usado.
- `table_name`: o nome da tabela a ser usada.
- `options`: opções para filtrar arquivos a serem excluídos e para geração do arquivo manifesto.

- `retentionPeriod`: especifica um período em quantidade de horas para reter arquivos. Os arquivos mais recentes do que o período de retenção serão mantidos. Definido como 168 horas (7 dias) por padrão.
- `partitionPredicate`: as partições que satisfazem esse predicado são excluídas. Os arquivos dentro do período de retenção nessas partições não são excluídos. Definido como `""`, vazio por padrão.
- `excludeStorageClasses`: os arquivos com classe de armazenamento no conjunto `excludeStorageClasses` não são excluídos. O padrão é `Set()`, um conjunto vazio.
- `manifestFilePath`: um caminho opcional para a geração do arquivo manifesto. Todos os arquivos que foram removidos com êxito são registrados no `Success.csv`, e os que falharam no `Failed.csv`

- `transformation_ctx` – O contexto de transformação a ser usado (opcional). Usado no caminho do arquivo de manifesto.

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="")`

Exclui arquivos do caminho Amazon S3 especificado recursivamente.

Se quiser recuperar objetos excluídos, você pode habilitar o versionamento de objeto no bucket do Amazon S3. Quando um objeto é excluído de um bucket que não tem o versionamento de objeto habilitado, ele não pode ser recuperado. Para obter mais informações sobre como recuperar objetos excluídos em um bucket com controle de versão, consulte Como posso recuperar um objeto do Amazon S3 que foi excluído? no Centro de AWS Support Conhecimento.

- `s3_path`: o caminho no Amazon S3 dos arquivos a serem excluídos no formato `s3://<bucket>/<prefix>/`
- `options`: opções para filtrar arquivos a serem excluídos e para geração do arquivo manifesto.

- `retentionPeriod`: especifica um período em quantidade de horas para reter arquivos. Os arquivos mais recentes do que o período de retenção serão mantidos. Definido como 168 horas (7 dias) por padrão.
- `excludeStorageClasses`: os arquivos com classe de armazenamento no conjunto `excludeStorageClasses` não são excluídos. O padrão é `Set()`, um conjunto vazio.
- `manifestFilePath`: um caminho opcional para a geração do arquivo manifesto. Todos os arquivos que foram removidos com êxito são registrados no `Success.csv`, e os que falharam no `Failed.csv`

- `transformation_ctx` – O contexto de transformação a ser usado (opcional). Usado no caminho do arquivo de manifesto.

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)`

Move a classe de armazenamento dos arquivos armazenados no Amazon S3 para o banco de dados e tabela do catálogo especificados.

É possível fazer a transição entre duas classes de armazenamento quaisquer. Nas classes de armazenamento `DEEP_ARCHIVE` e `GLACIER` e, você pode fazer a transição para essas classes. No entanto, você usaria um `S3 RESTORE` para fazer a transição das classes de armazenamento `GLACIER` e `DEEP_ARCHIVE`.

Se você estiver executando trabalhos de ETL do AWS Glue que leiam arquivos ou partições do Amazon S3, pode excluir alguns tipos de classe de armazenamento do Amazon S3. Para obter mais informações, consulte Excluir classes de armazenamento do Amazon S3.

- `database` – O mecanismo de banco de dados a ser usado.
- `table_name`: o nome da tabela a ser usada.
- `transition_to`: a classe de armazenamento do Amazon S3 para onde mover.

- `options`: opções para filtrar arquivos a serem excluídos e para geração do arquivo manifesto.
  - `retentionPeriod`: especifica um período em quantidade de horas para reter arquivos. Os arquivos mais recentes do que o período de retenção serão mantidos. Definido como 168 horas (7 dias) por padrão.
  - `partitionPredicate`: as partições que satisfazem esse predicado são movidas. Os arquivos dentro do período de retenção nessas partições não são transicionados. Definido como `""`, vazio por padrão.
  - `excludeStorageClasses`: os arquivos com classe de armazenamento no conjunto `excludeStorageClasses` não são movidos. O padrão é `Set()`, um conjunto vazio.
  - `manifestFilePath`: um caminho opcional para a geração do arquivo manifesto. Todos os arquivos que foram transicionados com êxito são registrados no `Success.csv`, e os que falharam no `Failed.csv`
  - `accountId`: o ID da conta da Amazon Web Services para executar a transformação de transição. Obrigatório para essa transformação.
  - `roleArn`— O AWS papel de executar a transformação da transição. Obrigatório para essa transformação.
- `transformation_ctx` – O contexto de transformação a ser usado (opcional). Usado no caminho do arquivo de manifesto.
- `catalog_id`: o ID do catálogo do Data Catalog sendo acessado (o ID da conta do Data Catalog). Definido como `None` por padrão. O `None` é definido como padrão para o ID do catálogo da conta de chamada no serviço.

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="")`

Move a classe de armazenamento dos arquivos no caminho do Amazon S3 especificado recursivamente.

É possível fazer a transição entre duas classes de armazenamento quaisquer. Nas classes de armazenamento `DEEP_ARCHIVE` e `GLACIER` e, você pode fazer a transição para essas classes. No entanto, você usaria um `S3 RESTORE` para fazer a transição das classes de armazenamento `GLACIER` e `DEEP_ARCHIVE`.

Se você estiver executando trabalhos de ETL do AWS Glue que leiam arquivos ou partições do Amazon S3, pode excluir alguns tipos de classe de armazenamento do Amazon S3. Para obter mais informações, consulte Excluir classes de armazenamento do Amazon S3.

- `s3_path`: o caminho no Amazon S3 dos arquivos a serem movidos no formato `s3://<bucket>/<prefix>/`
- `transition_to`: a classe de armazenamento do Amazon S3 para onde mover.
- `options`: opções para filtrar arquivos a serem excluídos e para geração do arquivo manifesto.
  - `retentionPeriod`: especifica um período em quantidade de horas para reter arquivos. Os arquivos mais recentes do que o período de retenção serão mantidos. Definido como 168 horas (7 dias) por padrão.
  - `partitionPredicate`: as partições que satisfazem esse predicado são movidas. Os arquivos dentro do período de retenção nessas partições não são transicionados. Definido como `""`, vazio por padrão.
  - `excludeStorageClasses`: os arquivos com classe de armazenamento no conjunto `excludeStorageClasses` não são movidos. O padrão é `Set()`, um conjunto vazio.
  - `manifestFilePath`: um caminho opcional para a geração do arquivo manifesto. Todos os arquivos que foram transicionados com êxito são registrados no `Success.csv`, e os que falharam no `Failed.csv`
  - `accountId`: o ID da conta da Amazon Web Services para executar a transformação de transição. Obrigatório para essa transformação.
  - `roleArn`— O AWS papel de executar a transformação da transição. Obrigatório para essa transformação.
- `transformation_ctx` – O contexto de transformação a ser usado (opcional). Usado no caminho do arquivo de manifesto.

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"})
```

Extração

- extract_jdbc_conf

extract_jdbc_conf

`extract_jdbc_conf(connection_name, catalog_id = None)`

Retorna um `dict` com chaves com as propriedades de configuração do objeto de conexão da AWS Glue no catálogo de dados.

- `user`: o nome de usuário do banco de dados.
- `password`: a senha do banco de dados.
- `vendor`: especifica um fornecedor (`mysql`, `postgresql`, `oracle`, `sqlserver` etc.).
- `enforceSSL`: uma string booleana indicando se é necessária uma conexão segura.
- `customJDBCCert`: use um certificado de cliente específico no caminho do Amazon S3 indicado.
- `skipCustomJDBCCertValidation`: uma string booleana indicando se o `customJDBCCert` deve ser validado por uma autoridade de certificação.
- `customJDBCCertString`: informações adicionais sobre o certificado personalizado, específicas para o tipo de driver.
- `url`: (obsoleto) URL do JDBC apenas com protocolo, servidor e porta.
- `fullUrl`: URL do JDBC como inserido quando a conexão foi criada (disponível no AWS Glue versão 3.0 ou posterior).

Exemplo de recuperação de configurações do 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'}
```

Transações

- start_transaction
- commit_transaction
- cancel_transaction

start_transaction

`start_transaction(read_only)`

Iniciar uma nova transação. Chama internamente a API startTransaction do Lake Formation.

- `read_only`: (booleano) indica se esta transação deve ser somente de leitura ou de leitura e gravação. As gravações feitas usando um ID de transação somente de leitura serão rejeitadas. As transações somente de leitura não precisam ser confirmadas.

Retorna o ID da transação.

commit_transaction

`commit_transaction(transaction_id, wait_for_commit = True)`

Tenta confirmar a transação especificada. `commit_transaction` pode retornar antes que a transação tenha terminado de confirmar. Chama internamente a API commitTransaction do Lake Formation.

- `transaction_id` : (string) a transação a ser confirmada.
- `wait_for_commit`: (booleano) Determina se `commit_transaction` retorna imediatamente. O valor padrão é true. Se for falso, `commit_transaction` sonda e aguarda até que a transação seja confirmada. A quantidade de tempo de espera é restrita a 1 minuto usando recuo exponencial com um máximo de 6 tentativas.

Retorna um booleano para indicar se a confirmação foi feita ou não.

cancel_transaction

`cancel_transaction(transaction_id)`

Tenta cancelar a transação especificada. Retorna uma exceção `TransactionCommittedException` se a transação tiver sido confirmada anteriormente. Chama internamente a CancelTransactionAPI Lake Formation.

- `transaction_id`: (string) a transação a ser cancelada.

Writing

- 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)`

Obtém um objeto `DataSink` que pode ser usado para escrever `DynamicFrames` em fontes externas. Verifique o `format` do SparkSQL primeiro para garantir a obtenção do depósito esperado.

- `connection_type`: o tipo de conexão a ser usado, como Amazon S3, Amazon Redshift e JDBC. Os valores válidos incluem `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle`, `kinesis` e `kafka`.
- `format` – o formato do SparkSQL a ser usado (opcional).
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `options`: uma coleção de pares nome-valor usados para especificar as opções de conexão. Alguns dos valores possíveis são:
  - `user` e `password`:pPara autorização
  - `url`:o endpoint para o datastore

- `dbtable`: o nome da tabela de destino.
- `bulkSize`: grau de paralelismo para operações de inserção

As opções que você pode especificar dependem do tipo de conexão. Consulte Tipos e opções de conexão para ETL no AWS Glue para Spark para obter valores e exemplos adicionais.

Exemplo:

```
>>> 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 = "")`

Escreve e retorna um `DynamicFrame` usando a conexão e o formato especificados.

- `frame` – O `DynamicFrame` a ser escrito.
- `connection_type`: o tipo de conexão, por exemplo, Amazon S3, Amazon Redshift e JDBC. Os valores válidos incluem `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle`, `kinesis` e `kafka`.
- `connection_options` – Opções de conexão, como caminho e tabela de banco de dados (opcional). Para um `connection_type` do `s3`, um caminho do Amazon S3 é definido.

  ```
  connection_options = {"path": "s3://aws-glue-target/temp"}
  ```

  Para conexões JDBC, várias propriedades devem ser definidas. Observe que o nome do banco de dados deve fazer parte do URL. Ele também pode ser incluído nas opções de conexão.

  > **Warning**
  >
  > Não é recomendável armazenar senhas no script. Considere usar `boto3` para recuperá-los do AWS Secrets Manager ou do AWS Glue Data Catalog.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

A propriedade `dbtable` é o nome da tabela JDBC. Para armazenamentos de dados JDBC que oferecem suporte a esquemas dentro de um banco de dados, especifique `schema.table-name`. Se um esquema não for fornecido, o esquema "público" padrão será usado.

Para ter mais informações, consulte Tipos e opções de conexão para ETL no AWS Glue para Spark.

- `format`— Uma especificação de formato. É usado para uma conexão do Amazon S3 ou do AWS Glue com suporte para vários formatos. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `format_options`: as opções de formato para o formato especificado. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `transformation_ctx` – Um contexto de transformação a ser usado (opcional).

write_from_options

`write_from_options(frame_or_dfc, connection_type, connection_options={}, format={}, format_options={}, transformation_ctx = "")`

Escreve e retorna o código `DynamicFrame` ou `DynamicFrameCollection` criado com as informações de conexão e formato especificadas.

- `frame_or_dfc` – O código `DynamicFrame` ou `DynamicFrameCollection` a ser escrito.
- `connection_type`: o tipo de conexão, por exemplo, Amazon S3, Amazon Redshift e JDBC. Os valores válidos incluem `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` e `oracle`.
- `connection_options` – Opções de conexão, como caminho e tabela de banco de dados (opcional). Para um `connection_type` do `s3`, um caminho do Amazon S3 é definido.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

Para conexões JDBC, várias propriedades devem ser definidas. Observe que o nome do banco de dados deve fazer parte do URL. Ele também pode ser incluído nas opções de conexão.

> **Warning**
>
> Não é recomendável armazenar senhas no script. Considere usar `boto3` para recuperá-los do AWS Secrets Manager ou do AWS Glue Data Catalog.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

A propriedade `dbtable` é o nome da tabela JDBC. Para armazenamentos de dados JDBC que oferecem suporte a esquemas dentro de um banco de dados, especifique `schema.table-name`. Se um esquema não for fornecido, o esquema "público" padrão será usado.

Para ter mais informações, consulte Tipos e opções de conexão para ETL no AWS Glue para Spark.

- `format`— Uma especificação de formato. É usado para uma conexão do Amazon S3 ou do AWS Glue com suporte para vários formatos. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `format_options`: as opções de formato para o formato especificado. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para obter os formatos compatíveis.
- `transformation_ctx` – Um contexto de transformação a ser usado (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)`

Grava e retorna um `DynamicFrame` usando um banco de dados e uma tabela do Data Catalog.

- `frame` – O `DynamicFrame` a ser escrito.

- `Database`: o banco de dados do Data Catalog que contém a tabela.
- `table_name`: o nome da tabela do Data Catalog associada ao destino.
- `redshift_tmp_dir`: um diretório temporário do Amazon RedShift a ser usado (opcional).
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `additional_options` – Uma coleção de pares nome-valor opcionais.
- `catalog_id`: o ID do catálogo (ID da conta) do Data Catalog que está sendo acessado. Quando None (Nenhum), o ID da conta do chamador padrão é usado.

write_data_frame_from_catalog

`write_data_frame_from_catalog(frame, database, table_name, redshift_tmp_dir, transformation_ctx = "", additional_options = {}, catalog_id = None)`

Grava e retorna um `DataFrame` usando um banco de dados e uma tabela do Data Catalog. Esse método é compatível com a gravação nos formatos de data lake (Hudi, Iceberg e Delta Lake). Para ter mais informações, consulte Usar estruturas de data lake com trabalhos do AWS Glue ETL.

- `frame` – O `DataFrame` a ser escrito.
- `Database`: o banco de dados do Data Catalog que contém a tabela.
- `table_name`: o nome da tabela do Data Catalog associada ao destino.
- `redshift_tmp_dir`: um diretório temporário do Amazon Redshift a ser usado (opcional).
- `transformation_ctx` – O contexto de transformação a ser usado (opcional).
- `additional_options` – Uma coleção de pares nome-valor opcionais.
  - `useSparkDataSink`— Quando definido como true, força o AWS Glue a usar a API nativa do Spark Data Sink para gravar na tabela. Ao ativar essa opção, você pode adicionar qualquer opção de fonte de dados do Spark `additional_options` conforme necessário. AWS Glue passa essas opções diretamente para o gravador do Spark.
- `catalog_id`: o ID do catálogo (ID da conta) do Data Catalog que está sendo acessado. Quando você não especifica um valor, será usado o ID padrão da conta do chamador.

Limitações

Considere as seguintes limitações ao usar a opção `useSparkDataSink`:

- A opção enableUpdateCatalog não é compatível quando você usa a opção useSparkDataSink.

Exemplo: gravação em uma tabela Hudi usando o gravador do Spark Data Source

```
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)

Escreve e retorna um DynamicFrame usando as informações de conexão JDBC especificadas.

- frame – O DynamicFrame a ser escrito.
- catalog_connection – Uma conexão de catálogo a ser usada.

- `connection_options` – Opções de conexão, como caminho e tabela de banco de dados (opcional). Para ter mais informações, consulte Tipos e opções de conexão para ETL no AWS Glue para Spark.
- `redshift_tmp_dir`: um diretório temporário do Amazon RedShift a ser usado (opcional).
- `transformation_ctx` – Um contexto de transformação a ser usado (opcional).
- `catalog_id`: o ID do catálogo (ID da conta) do Data Catalog que está sendo acessado. Quando None (Nenhum), o ID da conta do chamador padrão é usado.

write_from_jdbc_conf

`write_from_jdbc_conf(frame_or_dfc, catalog_connection, connection_options={}, redshift_tmp_dir = "", transformation_ctx = "", catalog_id = None)`

Escreve e retorna um código `DynamicFrame` ou `DynamicFrameCollection` usando as informações de conexão JDBC especificadas.

- `frame_or_dfc` – O código `DynamicFrame` ou `DynamicFrameCollection` a ser escrito.
- `catalog_connection` – Uma conexão de catálogo a ser usada.
- `connection_options` – Opções de conexão, como caminho e tabela de banco de dados (opcional). Para ter mais informações, consulte Tipos e opções de conexão para ETL no AWS Glue para Spark.
- `redshift_tmp_dir`: um diretório temporário do Amazon RedShift a ser usado (opcional).
- `transformation_ctx` – Um contexto de transformação a ser usado (opcional).
- `catalog_id`: o ID do catálogo (ID da conta) do Data Catalog que está sendo acessado. Quando None (Nenhum), o ID da conta do chamador padrão é usado.

AWS Glue PySpark transforma a referência

AWS O Glue fornece as seguintes transformações integradas que você pode usar em operações de PySpark ETL. Seus dados passam de transformação em transformação em uma estrutura de dados chamada a DynamicFrame, que é uma extensão de um Apache Spark SQL. `DataFrame` O `DynamicFrame` contém os dados, e você referencia o esquema para processar os dados.

A maioria dessas transformações também existe como métodos da classe `DynamicFrame`. Para obter mais informações, consulte DynamicFrame transformações.

Classe de base GlueTransform

A classe base que todas as classes `awsglue.transforms` herdam.

Todas as classes definem um método `__call__`. Elas substituem os métodos da classe `GlueTransform` listados nas seções a seguir ou são chamadas usando o nome de classe, por padrão.

Métodos

- apply(cls, *args, **kwargs)
- name(cls)
- describeArgs(cls)
- describeReturn(cls)
- describeTransform(cls)
- describeErrors(cls)
- describe(cls)

apply(cls, *args, **kwargs)

Aplica a transformação chamando a classe de transformação e, em seguida, retorna o resultado.

- `cls` – O objeto de classe `self`.

name(cls)

Retorna o nome da classe de transformação derivada.

- `cls` – O objeto de classe `self`.

describeArgs(cls)

- `cls` – O objeto de classe `self`.

Retorna uma lista de dicionários, cada um correspondente a um argumento nomeado, no seguinte 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)"
  },
...
]
```

Gera uma exceção `NotImplementedError` quando chamado em uma transformação derivada em que não foi implementado.

describeReturn(cls)

- `cls` – O objeto de classe `self`.

Retorna um dicionário com informações sobre o tipo de retorno, no seguinte formato:

```
{
  "type": "(return type)",
  "description": "(description of output)"
}
```

Gera uma exceção `NotImplementedError` quando chamado em uma transformação derivada em que não foi implementado.

describeTransform(cls)

Retorna uma string que descreve a transformação.

- `cls` – O objeto de classe `self`.

Gera uma exceção `NotImplementedError` quando chamado em uma transformação derivada em que não foi implementado.

describeErrors(cls)

- `cls` – O objeto de classe `self`.

Retorna uma lista de dicionários, cada um descrevendo uma possível exceção lançada pela transformação, no seguinte formato:

```
[
  {
    "type": "(type of error)",
    "description": "(description of error)"
  },
...
]
```

describe(cls)

- `cls` – O objeto de classe `self`.

Retorna um objeto com o seguinte formato:

```
{
  "transform" : {
    "name" : cls.name( ),
    "args" : cls.describeArgs( ),
    "returns" : cls.describeReturn( ),
    "raises" : cls.describeErrors( ),
    "location" : "internal"
  }
}
```

Classe ApplyMapping

Aplica um mapeamento em um `DynamicFrame`.

Exemplo

Recomendamos usar o método `DynamicFrame.apply_mapping()` para aplicar um mapeamento em um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar apply_mapping para renomear campos e alterar os tipos de campo.

Métodos

- __call__
- apply
- name
- describeArgs

- describeReturn
- describeTransform
- describeErrors
- Descrever

__call__(frame, mappings, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Aplica um mapeamento declarativo a um `DynamicFrame` especificado.

- `frame`: o `DynamicFrame` em que o mapeamento será aplicado (obrigatório).
- `mappings`: uma lista de tuplas de mapeamento (obrigatório). Cada uma é composta por: coluna de fonte, tipo de fonte, coluna de destino, tipo de destino.

  Se a coluna de origem tiver um ponto "`.`" no nome, será necessário colocar crases "```" em torno dela. Por exemplo, para mapear `this.old.name` (string) para `thisNewName`, você pode usar a seguinte tupla:

  ```
  ("`this.old.name`", "string", "thisNewName", "string")
  ```
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info`: uma cadeira de caracteres que é associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

Retorna somente os campos do `DynamicFrame` que são especificados nas tuplas de "mapeamento".

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe DropFields

Descartar campos em um `DynamicFrame`.

Exemplo

Recomendamos usar o método `DynamicFrame.drop_fields()` para descartar campos de um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar drop_fields para remover campos de um `DynamicFrame`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Descrever

__call__(frame, paths, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Colocar nós em um `DynamicFrame`.

- `frame`: o `DynamicFrame` onde os nós serão descartados (obrigatório).
- `paths` – Uma lista de caminhos completos para os nós a serem descartados (obrigatório).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

Retorna um novo `DynamicFrame` sem os campos especificados.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe DropNullFields

Descarta todos os campos nulos em um `DynamicFrame` cujos tipos são `NullType`. São campos com valores ausentes ou nulos em todos os registros no conjunto de dados `DynamicFrame`.

Exemplo

Este exemplo usa `DropNullFields` para criar um novo `DynamicFrame` no qual os campos do tipo `NullType` foram descartados. Para demonstrar `DropNullFields`, adicionamos uma nova coluna do tipo null chamada `empty_column` ao conjunto de dados `persons` já carregado.

> **Note**
>
> Para acessar o conjunto de dados usado neste exemplo, consulte Exemplo de código: juntar e relacionar dados e siga as instruções em Etapa 1: crawling de dados no bucket do 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()
```

Saída

```
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
- Descrever

__call__(frame, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Descarta todos os campos nulos em um `DynamicFrame` cujos tipos são `NullType`. São campos com valores ausentes ou nulos em todos os registros no conjunto de dados `DynamicFrame`.

- `frame`: `DynamicFrame` onde os campos nulos serão descartados (obrigatório).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

Retorna um novo `DynamicFrame` sem campos nulos.

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

Classe ErrorsAsDynamicFrame

Retorna um `DynamicFrame` que contém registros aninhados de erros que ocorreram durante a criação da origem `DynamicFrame`.

Exemplo

Recomendamos usar o método `DynamicFrame.errorsAsDynamicFrame()` para recuperar e visualizar registros de erros. Para visualizar um código de exemplo, consulte Exemplo: usar errorsAsDynamicFrame para visualizar registros de erros.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Descrever

__call__(frame)

Retorna um `DynamicFrame` que contém registros de erros aninhados que estão relacionados ao `DynamicFrame` de origem.

- `frame` – O `DynamicFrame` de origem (obrigatório).

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

Classe EvaluateDataQuality

Avalia um conjunto de regras de qualidade de dados em relação a um `DynamicFrame` e retorna um novo `DynamicFrame` com os resultados da avaliação.

Exemplo

O código de exemplo a seguir demonstra como avaliar a qualidade dos dados para um `DynamicFrame` e, em seguida, visualizar os resultados de qualidade dos dados.

```
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()
```

Saída

```
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`: o `DynamicFrame` cuja qualidade de dados você deseja avaliar.
- `ruleset`: um conjunto de regras em Data Quality Definition Language (DQDL) no formato de string. Para saber mais sobre DQDL, consulte o guia de Referência de Data Quality Definition Language (DQDL).
- `publishing_options`: um dicionário que especifica as seguintes opções para publicar resultados e métricas de avaliação:
  - `dataQualityEvaluationContext`: uma string que especifica o namespace com o qual o AWS Glue deve publicar as métricas e os resultados de qualidade dos dados do Amazon CloudWatch. As métricas agregadas aparecem no CloudWatch, enquanto os resultados completos aparecem na interface do AWS Glue Studio.
    - Obrigatório: Não
    - Valor padrão: `default_context`

- `enableDataQualityCloudWatchMetrics`: especifica se os resultados da avaliação de qualidade dos dados devem ser publicados no CloudWatch. Você especifica um namespace para as métricas usando a opção `dataQualityEvaluationContext`.
  - Obrigatório: Não
  - Valor padrão: False
- `enableDataQualityResultsPublishing`: especifica se os resultados de qualidade dos dados devem estar visíveis na guia Data Quality (Qualidade de dados) na interface do AWS Glue Studio.
  - Obrigatório: Não
  - Valor padrão: true
- `resultsS3Prefix`: especifica o local no Amazon S3 em que o AWS Glue pode gravar os resultados da avaliação de qualidade dos dados.
  - Obrigatório: Não
  - Valor padrão: "" (string vazia)

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe FillMissingValues

A classe `FillMissingValues` localiza valores null e strings vazias em um `DynamicFrame` especificado e usa métodos de machine learning, como regressão linear e floresta aleatória, para prever os valores ausentes. O trabalho de ETL usa os valores no conjunto de dados de entrada para treinar o modelo de machine learning, que então prevê quais devem ser os valores ausentes.

> **Tip**
>
> Se você usar conjuntos de dados incrementais, cada conjunto incremental será usado como dados de treinamento para o modelo de machine learning, portanto, os resultados podem não ser tão precisos.

Para importar:

```
from awsglueml.transforms import FillMissingValues
```

Métodos

- Aplicar

apply(frame, missing_values_column, output_column ="", transformation_ctx ="", info ="", stageThreshold = 0, totalThreshold = 0)

Preenche os valores ausentes de um quadro dinâmico em uma coluna especificada e retorna um novo quadro com estimativas em uma nova coluna. Para linhas sem valores ausentes, o valor da coluna especificada é duplicado para a nova coluna.

- `frame`: o `DynamicFrame` no qual preencher valores ausentes. Obrigatório.
- `missing_values_column`: a coluna que contém valores ausentes (valores `null` e strings vazias). Obrigatório.
- `output_column`: o nome da nova coluna que conterá valores estimados para todas as linhas cujo valor estava ausente. Opcional; o padrão é o nome da `missing_values_column` seguida do sufixo `"_filled"`.

- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold` – O número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional, o padrão é zero).
- `totalThreshold` – O número máximo de erros que podem ocorrer antes que o processamento falhe (opcional, o padrão é zero).

Retorna um novo `DynamicFrame` com uma coluna adicional que contém estimativas para linhas com valores ausentes e o valor presente para outras linhas.

Classe Filter

Cria um novo `DynamicFrame` que contém registros do `DynamicFrame` de entrada que satisfazem uma função predicada especificada.

Exemplo

Recomendamos usar o método `DynamicFrame.filter()` para filtrar registros em um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar filter para obter uma seleção filtrada de campos.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0))

Retorna um novo `DynamicFrame` que é criado selecionando registros do `DynamicFrame` de entrada que satisfazem uma função predicada especificada.

- `frame` – O `DynamicFrame` de origem ao qual a função de filtro especificada será aplicada (obrigatório).
- `f`: a função predicada a ser aplicada a cada `DynamicRecord` no `DynamicFrame`. A função precisa ter um `DynamicRecord` como argumento e retornar True, se `DynamicRecord` atender aos requisitos de filtro, ou False, se não atender (obrigatório).

  Um `DynamicRecord` representa um registro lógico em um `DynamicFrame`. É semelhante a uma linha em um `DataFrame` do Spark, exceto pelo fato de que pode se autodescrever e ser usado para dados que não estejam em conformidade com um esquema fixo.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info`: uma cadeira de caracteres que é associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe FindIncrementalMatches

Identifica registros correspondentes no `DynamicFrame` incremental e existente e cria um novo `DynamicFrame` com um identificador exclusivo atribuído a cada grupo de registros correspondentes.

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 correspondentes no `DynamicFrame` de entrada e cria um novo `DynamicFrame` com um identificador exclusivo atribuído a cada grupo de registros correspondentes.

- `existingFrame`: o `DynamicFrame` existente e pré-correspondido em que aplicar a transformação FindIncrementalMatches. Obrigatório.
- `incrementalFrame`: o `DynamicFrame` incremental em que aplicar a transformação FindIncrementalMatches para corresponder a `existingFrame`. Obrigatório.
- `transformId`: o ID exclusivo associado à transformação FindIncrementalMatches a ser aplicado em registros no `DynamicFrames`. Obrigatório.
- `transformation_ctx`: uma string única que é usada para identificar informações de estado/ estatística. Opcional.
- `info`: uma string a ser associada a erros na transformação. Opcional.
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe. Opcional. O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe. Opcional. O padrão é zero.

- `enforcedMatches`: o `DynamicFrame` usado para impor correspondências. Opcional. O padrão é Nenhum.
- `computeMatchConfidenceScores`: um valor booleano que indica se a pontuação de confiança deve ser calculada para cada grupo de registros correspondentes. Opcional. O padrão é falso.

Retorna um novo `DynamicFrame` com um identificador exclusivo atribuído a cada grupo de registros correspondentes.

Classe FindMatches

Identifica registros correspondentes no `DynamicFrame` de entrada e cria um novo `DynamicFrame` com um identificador exclusivo atribuído a cada grupo de registros correspondentes.

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 correspondentes no `DynamicFrame` de entrada e cria um novo `DynamicFrame` com um identificador exclusivo atribuído a cada grupo de registros correspondentes.

- `frame`: o `DynamicFrame` em que aplicar a transformação FindMatches. Obrigatório.
- `transformId`: o ID exclusivo associado à transformação FindMatches a ser aplicado em registros no `DynamicFrame`. Obrigatório.
- `transformation_ctx`: uma string única que é usada para identificar informações de estado/ estatística. Opcional.
- `info`: uma string a ser associada a erros na transformação. Opcional.
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe. Opcional. O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe. Opcional. O padrão é zero.

- `enforcedMatches`: o `DynamicFrame` usado para impor correspondências. Opcional. O padrão é Nenhum.
- `computeMatchConfidenceScores`: um valor booleano que indica se a pontuação de confiança deve ser calculada para cada grupo de registros correspondentes. Opcional. O padrão é falso.

Retorna um novo `DynamicFrame` com um identificador exclusivo atribuído a cada grupo de registros correspondentes.

Classe FlatMap

Aplica uma transformação a cada `DynamicFrame` do conjunto. Os resultados não são reduzidos a um único `DynamicFrame`, mas preservados como um conjunto.

Exemplos de FlatMap

O trecho de exemplo a seguir demonstra como usar a transformação `ResolveChoice` em um conjunto de quadros dinâmicos quando aplicada a um `FlatMap`. Os dados usados para entrada estão no JSON localizado no endereço Amazon S3 reservado `s3://bucket/path-for-data/sample.json` e contêm os dados a seguir.

Exemplo de dados 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 uma DynamicFrameCollection e mostre a saída.

```
#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**
>
> Ao chamar `FlatMap.apply`, o parâmetro `frame_name` deve ser `"frame"`. Nenhum outro valor é aceito atualmente.

Exemplo de saída

```
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
- Nome
- describeArgs
- describeReturn

- describeTransform
- describeErrors
- Descrever

__call__(dfc, BaseTransform, frame_name, transformation_ctx = "", **base_kwargs)

Aplica uma transformação para cada `DynamicFrame` em uma coleção e nivela os resultados.

- `dfc` – O `DynamicFrameCollection` em que o flatmap será aplicado (obrigatório).
- `BaseTransform` – Uma transformação de `GlueTransform` a ser aplicada a cada membro da coleção (obrigatório).
- `frame_name` – O nome do argumento para o qual os elementos da coleção serão transmitidos (obrigatório).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `base_kwargs` – Os argumentos a serem transmitidos à transformação base (obrigatório).

Retorna um novo `DynamicFrameCollection` criado aplicando a transformação a cada `DynamicFrame` na origem `DynamicFrameCollection`.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe Join

Executa uma junção de igualdade em dois `DynamicFrames`.

Exemplo

Recomendamos usar o método `DynamicFrame.join()` para juntar `DynamicFrames`. Para visualizar um código de exemplo, consulte Exemplo: usar join para combinar `DynamicFrames`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Descrever

__call__(frame1, frame2, keys1, keys2, transformation_ctx = "")

Executa uma junção de igualdade em dois `DynamicFrames`.

- `frame1` – O primeiro `DynamicFrame` da junção (obrigatório).
- `frame2` – O segundo `DynamicFrame` da junção (obrigatório).
- `keys1` – As chaves para junção do primeiro quadro (obrigatório).
- `keys2` – As chaves para junção do segundo quadro (obrigatório).

- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).

Retorna um novo `DynamicFrame`, que é criado juntando os dois `DynamicFrames`.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe Map

Cria um novo `DynamicFrame` aplicando uma função a todos os registros no `DynamicFrame` de entrada.

Exemplo

Recomendamos usar o método `DynamicFrame.map()` para aplicar uma função a todos os registros em um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar map para aplicar uma função a cada registro em um `DynamicFrame`.

Métodos

__call__(frame, f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

Retorna um novo `DynamicFrame` que resulta da aplicação da função especificada a todos os `DynamicRecords` no `DynamicFrame` original.

- `frame`: o `DynamicFrame` original ao qual a função de mapeamento deverá ser aplicada (obrigatório).
- `f`: a função a ser aplicada a todos os `DynamicRecords` no `DynamicFrame`. A função precisa usar um `DynamicRecord` como argumento e retornar um novo `DynamicRecord` que seja produzido pelo mapeamento (obrigatório).

  Um `DynamicRecord` representa um registro lógico em um `DynamicFrame`. É semelhante a uma linha em um `DataFrame` do Apache Spark, exceto pelo fato de que pode se autodescrever e ser usado para dados que não estejam em conformidade com um esquema fixo.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

Retorna um novo `DynamicFrame` que resulta da aplicação da função especificada a todos os `DynamicRecords` no `DynamicFrame` original.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe MapToCollection

Aplica uma transformação a cada `DynamicFrame` no `DynamicFrameCollection` especificado.

Métodos

- __call__
- Aplicar
- Nome
- describeArgs
- describeReturn
- describeTransform
- describeErrors

- Descrever

__call__(dfc, BaseTransform, frame_name, transformation_ctx = "", **base_kwargs)

Aplica uma função de transformação a cada `DynamicFrame` no `DynamicFrameCollection` especificado.

- `dfc` – O `DynamicFrameCollection` no qual a função de transformação deverá ser aplicada (obrigatório).
- `callable` – Uma função de transformação que pode ser chamada para ser aplicada a cada membro da coleção (obrigatório).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).

Retorna um novo `DynamicFrameCollection` criado aplicando a transformação a cada `DynamicFrame` na origem `DynamicFrameCollection`.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe Relationalize

Nivela um esquema aninhado em um `DynamicFrame` e cria colunas dinâmicas de matriz a partir do quadro nivelado.

Exemplo

Recomendamos usar o método `DynamicFrame.relationalize()` para relacionalizar um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar relationalize para nivelar um esquema aninhado em um `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)

Relacionaliza um `DynamicFrame` e produz uma lista de quadros que são gerados desaninhando colunas aninhadas e criando colunas dinâmicas de matriz. Uma coluna de matriz dinâmica pode ser adicionada à tabela raiz usando a chave de união gerada durante a fase de desaninhamento.

- `frame`: o `DynamicFrame` a ser relacionalizado (obrigatório).
- `staging_path`: o caminho onde o método pode armazenar partições de tabelas dinâmicas no formato CSV (opcional). As tabelas dinâmicas são lidas novamente nesse caminho.
- `name` – O nome da tabela raiz (opcional).

- `options` – Um dicionário de parâmetros opcionais. Não utilizado no momento.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe RenameField

Renomeia um nó em um `DynamicFrame`.

Exemplo

Recomendamos usar o método `DynamicFrame.rename_field()` para renomear um campo em um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar rename_field para renomear campos em um `DynamicFrame`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, old_name, new_name, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Renomeia um nó em um `DynamicFrame`.

- `frame` – O `DynamicFrame` no qual o nó será renomeado (obrigatório).
- `old_name`: o caminho completo para o nó a ser renomeado (obrigatório).

  Se o nome antigo tiver pontos, o RenameField não funcionará a menos que você posicione os acentos graves (``) ao seu redor. Por exemplo, para substituir `this.old.name` por `thisNewName`, você chamaria RenameField, da seguinte maneira:

  ```
  newDyF = RenameField(oldDyF, "`this.old.name`", "thisNewName")
  ```

- `new_name`: o novo nome, incluindo o caminho completo (obrigatório).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.

- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe ResolveChoice

Resolve um tipo de escolha dentro de um `DynamicFrame`.

Exemplo

Recomendamos usar o método `DynamicFrame.resolveChoice()` para lidar com campos que contêm vários tipos em um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar resolveChoice para lidar com uma coluna que contém vários tipos.

Métodos

- __call__
- apply

- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, specs = none, choice = "", transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Fornece informações para resolver tipos ambíguos dentro de um `DynamicFrame`. Retorna o `DynamicFrame` resultante.

- `frame` – O `DynamicFrame` no qual o tipo de escolha será resolvido (obrigatório).
- `specs`: uma lista de ambiguidades específicas para resolver, cada uma na forma de uma tupla: `(path, action)`. O valor `path` identifica um elemento ambíguo específico, e o valor `action` identifica a resolução correspondente.

  É possível usar somente um dos parâmetros `spec` e `choice`. Se o parâmetro `spec` não for `None`, o parâmetro `choice` precisará ser uma string vazia. Por outro lado, se o `choice` não for uma string vazia, o parâmetro `spec` precisará ser `None`. Se nenhum dos parâmetros for fornecido, o AWS Glue tentará analisar o esquema e usá-lo para resolver ambiguidades.

  É possível especificar uma das seguintes estratégias de resolução na porção `action` de uma tupla `specs`:

  - `cast`: permite especificar um tipo a ser convertido (por exemplo: `cast:int`).
  - `make_cols`: resolve uma possível ambiguidade ao nivelar os dados. Por exemplo, se `columnA` puder ser `int` ou `string`, a resolução é produzir duas colunas chamadas `columnA_int` e `columnA_string` no `DynamicFrame` resultante.
  - `make_struct`: resolve uma ambiguidade em potencial usando um struct para representar os dados. Por exemplo, se os dados em uma coluna pudessem ser um `int` ou `string`, usar a ação `make_struct` produzirá uma coluna de estruturas no `DynamicFrame` resultante, com cada uma contendo `int` e `string`.
  - `project`: resolve uma possível ambiguidade retendo somente valores de um tipo especificado no `DynamicFrame` resultante Por exemplo, se os dados em uma coluna `ChoiceType`

pudessem ser um `int` ou uma `string`, especificar uma ação `project:string` descartaria valores do `DynamicFrame` resultante que não fossem do tipo `string`.

Se o `path` identifica um array, insira colchetes vazios após o nome do array para evitar ambiguidades. Por exemplo, vamos supor que você esteja trabalhando com dados estruturados da seguinte maneira:

```
"myList": [
  { "price": 100.00 },
  { "price": "$100.00" }
]
```

Você pode selecionar a versão numérica em vez da versão string do preço definindo `path` como `"myList[].price"` e `action` como `"cast:double"`.

- `choice`: a ação de resolução padrão se o parâmetro `specs` for `None`. Se o parâmetro `specs` não for `None`, isso deve ser configurado como uma string vazia.

  Além das ações `specs` listadas anteriormente, este modo também aceita a seguinte ação:

  - `MATCH_CATALOGChoiceType`: tenta converter cada para o tipo correspondente na tabela do Data Catalog especificada.
- `database`: o banco de dados do AWS Glue Data Catalog a ser usado com a escolha `MATCH_CATALOG` (obrigatório para `MATCH_CATALOG`).
- `table_name`: o nome da tabela do AWS Glue Data Catalog a ser usado com a ação `MATCH_CATALOG` (obrigatório para `MATCH_CATALOG`).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe SelectFields

A classe `SelectFields` cria um novo `DynamicFrame` com base em um `DynamicFrame` existente e mantém somente os campos que você especificar. `SelectFields` oferece uma funcionalidade semelhante a uma instrução SQL `SELECT`.

Exemplo

Recomendamos usar o método `DynamicFrame.select_fields()` para selecionar campos de um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar select_fields para criar um novo `DynamicFrame` com os campos escolhidos.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn

- describeTransform
- describeErrors
- Descrever

__call__(frame, paths, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Recebe campos (nós) em um `DynamicFrame`.

- `frame`: o `DynamicFrame` onde os campos serão selecionados (obrigatório).
- `paths` – Uma lista de caminhos completos para os campos a serem selecionados (obrigatório).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info`: uma cadeira de caracteres que é associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

Retorna um novo `DynamicFrame` que contém somente os campos especificados.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe SelectFromCollection

Seleciona um `DynamicFrame` em `DynamicFrameCollection`.

Exemplo

Este exemplo usa `SelectFromCollection` para selecionar um `DynamicFrame` de um `DynamicFrameCollection`.

Exemplo de conjunto de dados

O exemplo seleciona dois `DynamicFrames` de um `DynamicFrameCollection` chamado `split_rows_collection`. Está é uma lista de limites no `split_rows_collection`.

```
dict_keys(['high', 'low'])
```

Código de exemplo

```
# 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()
```

Saída

```
+---+-----+-----------------------+-----------------------+
| 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 = "")

Obtém um `DynamicFrame` de `DynamicFrameCollection`.

- `dfc`: a `DynamicFrameCollection` da qual o `DynamicFrame` deve ser selecionado (obrigatório).
- `key` – A chave do `DynamicFrame` a ser selecionada (obrigatório).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe Simplify_ddb_json

Simplifica colunas aninhadas em um `DynamicFrame` que estão especificamente na estrutura JSON do DynamoDB e retorna um novo `DynamicFrame` simplificado.

Exemplo

Recomendamos usar o método `DynamicFrame.simplify_ddb_json()` para simplificar colunas aninhadas em um `DynamicFrame` que estão especificamente na estrutura JSON do DynamoDB. Para visualizar um código de exemplo, consulte Exemplo: use simplify_ddb_json para invocar uma simplificação JSON do DynamoDB.

Classe Spigot

Grava registros de exemplo em um destino específico para ajudar você a verificar as transformações realizadas pelo seu trabalho da AWS Glue.

Exemplo

Recomendamos usar o método `DynamicFrame.spigot()` para gravar um subconjunto de registros de um `DynamicFrame` em um destino especificado. Para visualizar um código de exemplo,

consulte Exemplo: usar spigot para gravar campos de exemplo de um `DynamicFrame` no Amazon S3.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, path, options, transformation_ctx = "")

Grava registros de amostra em um destino especificado durante uma transformação.

- `frame` – O `DynamicFrame` para spigot (obrigatório).
- `path`: o caminho para o destino no qual a gravação será feita (obrigatório).
- `options`: pares de chave-valor JSON que especificam opções (opcional). A opção `"topk"` especifica que os primeiros registros k devem ser gravados. A opção `"prob"` especifica a probabilidade (como um número decimal) de selecionar um determinado registro. Isso pode ser usado na seleção de registros para gravar.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply

name(cls)

Herdado de `GlueTransform` name

describeArgs(cls)

Herdado de `GlueTransform` describeArgs

describeReturn(cls)

Herdado de `GlueTransform` describeReturn

describeTransform(cls)

Herdado de `GlueTransform` describeTransform

describeErrors(cls)

Herdado de `GlueTransform` describeErrors

describe(cls)

Herdado de `GlueTransform` describe

Classe SplitFields

Separa um `DynamicFrame` em dois novos, por campos especificados.

Exemplo

Recomendamos usar o método `DynamicFrame.split_fields()` para dividir campos em um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar split_fields para dividir os campos selecionados em um `DynamicFrame` separado.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, paths, name1 = none, name2 = none, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Separa um ou mais campos em um `DynamicFrame` em um novo `DynamicFrame` e cria outro novo `DynamicFrame` que contém os campos que permaneceram.

- `frame` – O `DynamicFrame` de origem a ser separado em dois novos (obrigatório).
- `paths` – Uma lista de caminhos completos para os campos a serem separados (obrigatório).
- `name1`: o nome a ser atribuído ao `DynamicFrame` que conterá os campos a serem divididos (opcional). Se nenhum nome for fornecido, o nome da estrutura de origem será usado com "1" anexado.
- `name2`: o nome a ser atribuído ao `DynamicFrame` que conterá os campos que permanecerão após aqueles especificados terem sido separados (opcional). Se nenhum nome for informado, o nome da estrutura de origem será usado com "2" anexado.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe SplitRows

Cria um `DynamicFrameCollection` que contém dois `DynamicFrames`. Um `DynamicFrame` contém apenas as linhas especificadas a serem separadas e o outro contém todas as linhas restantes.

Exemplo

Recomendamos usar o método `DynamicFrame.split_rows()` para dividir linhas em um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar split_rows para dividir linhas em um `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)

Separa uma ou mais linhas em um `DynamicFrame` em um novo `DynamicFrame`.

- `frame` – O `DynamicFrame` de origem a ser separado em dois novos (obrigatório).
- `comparison_dict`: um dicionário em que a chave é o caminho completo para uma coluna e o valor é outro dicionário para mapear comparadores a valores aos quais os valores das colunas são comparados. Por exemplo, `{"age": {">": 10, "<": 20}}` separa as linhas onde o valor de "age" está entre 10 e 20, incluindo as linhas nas quais "age" está fora do intervalo (obrigatório).
- `name1` – O nome a ser atribuído a `DynamicFrame` que conterá as linhas a serem separadas (opcional).
- `name2` – O nome a ser atribuído a `DynamicFrame` que conterá as linhas que permanecerão após as linhas especificadas terem sido separadas (opcional).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe Unbox

Descompacta (reformata) um campo de string em um `DynamicFrame`.

Exemplo

Recomendamos usar o método `DynamicFrame.unbox()` para descompactar um campo em um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar unbox para descompactar um campo de string em um struct.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, path, format, transformation_ctx = "", info="", stageThreshold=0, totalThreshold=0, **options)

Descompacta um campo de string em um `DynamicFrame`.

- `frame`: o `DynamicFrame` no qual um campo será descompactado (obrigatório).
- `path` – O caminho completo para o `StringNode` a ser descompactado (obrigatório).
- `format`: uma especificação de formato (opcional). É usado para uma conexão do Amazon S3 ou do AWS Glue que oferece suporte a vários formatos. Consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark para conhecer os formatos compatíveis.
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).

- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.
- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.
- `separator` – Um token de separação (opcional).
- `escaper` – Um token de escape (opcional).
- `skipFirst` – `True` se a primeira linha de dados tiver que ser ignorada ou `False` caso contrário (opcional).
- withSchema: uma string que contém o esquema de descompactação dos dados (opcional). Isso sempre deve ser criado usando `StructType.json`.
- `withHeader` – `True` se os dados descompactados incluírem um cabeçalho ou `False` caso contrário (opcional).

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe UnnestFrame

Desaninha um `DynamicFrame`, nivela objetos aninhados com elementos de nível superior e gera chaves de união para objetos de matriz.

Exemplo

Recomendamos usar o método `DynamicFrame.unnest()` para nivelar estruturas aninhadas em um `DynamicFrame`. Para visualizar um código de exemplo, consulte Exemplo: usar unnest para transformar campos aninhados em campos de nível superior.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, transformation_ctx = "", info="", stageThreshold=0, totalThreshold=0)

Desaninha um `DynamicFrame`, nivela objetos aninhados com elementos de nível superior e gera chaves de união para objetos de matriz.

- `frame` - O `DynamicFrame` a ser desaninhado (obrigatório).
- `transformation_ctx` – Uma string única que é usada para identificar informações de estado (opcional).
- `info` – Uma string associada a erros na transformação (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional). O padrão é zero.

- `totalThreshold`: o número máximo de erros que podem ocorrer em geral antes que falhe (opcional). O padrão é zero.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

FlagDuplicatesInColumn classe

A `FlagDuplicatesInColumn` transformação retorna uma nova coluna com um valor especificado em cada linha que indica se o valor na coluna de origem da linha corresponde a um valor em uma linha anterior da coluna de origem. Quando as correspondências são encontradas, elas são marcadas como duplicatas. A ocorrência inicial não é sinalizada porque não corresponde a uma linha anterior.

Exemplo

```
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
```

Saída

A `FlagDuplicatesInColumn` transformação adicionará uma nova coluna `flag_col` ao `df_output`. DataFrame Essa coluna conterá um valor de string indicando se a linha correspondente tem um valor duplicado na coluna `cidade` ou não. Se uma linha tiver um valor `cidade` duplicado, o `flag_col` conterá o valor `true_string` “True”. Se uma linha tiver um valor exclusivo de `cidade`, o `flag_col` conterá o valor `false_string` “False”.

O `df_output` resultante DataFrame conterá todas as colunas do `datasource1` original, mais a coluna `flag_col` adicional indicando valores duplicados de `cidade`. 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)

A `FlagDuplicatesInColumn` transformação retorna uma nova coluna com um valor especificado em cada linha que indica se o valor na coluna de origem da linha corresponde a um valor em uma linha anterior da coluna de origem. Quando as correspondências são encontradas, elas são marcadas como duplicatas. A ocorrência inicial não é sinalizada porque não corresponde a uma linha anterior.

- `source_column`— Nome da coluna de origem.
- `target_column`— Nome da coluna de destino.
- `true_string`— Cadeia de caracteres a ser inserida na coluna de destino quando o valor da coluna de origem duplica um valor anterior nessa coluna.
- `false_string`— Cadeia de caracteres a ser inserida na coluna de destino quando o valor da coluna de origem é diferente dos valores anteriores dessa coluna.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

FormatPhoneNumber classe

A `FormatPhoneNumber` transformação retorna uma coluna na qual uma string de número de telefone é convertida em um valor formatado.

Exemplo

```
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
```

Saída

A saída será:

```
```

```
+---------------+
| phone|
+---------------+
|(408) 341-5669|
|(408) 341-5669|
+---------------+
```

A `FormatPhoneNumber` transformação usa a `source_column` como `"phone"` e a `default_region` como `"US"`.

A transformação formata com sucesso os dois números de telefone, independentemente do formato inicial, no formato padrão dos EUA `(408) 341-5669`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_column, phone_number_format = Nenhum, default_region=Nenhum, default_region_column=Nenhum)

A `FormatPhoneNumber` transformação retorna uma coluna na qual uma string de número de telefone é convertida em um valor formatado.

- `source_column`: o nome de uma coluna existente.
- `phone_number_format`— O formato para o qual converter o número de telefone. Se nenhum formato for especificado, o padrão será um formato de número de telefone padrão reconhecido internacionalmente. `E.164` Entre os valores válidos estão os seguintes:
  - E164 (omitir o período após E)

- `default_region`— Um código de região válido que consiste em duas ou três letras maiúsculas que especifica a região do número de telefone quando nenhum código de país está presente no próprio número. No máximo, um dos `defaultRegion` ou `defaultRegionColumn` pode ser fornecido.
- `default_region_column`— O nome de uma coluna do tipo de dados avançado`Country`. O código da região da coluna especificada é usado para determinar o código do país para o número de telefone quando nenhum código de país está presente no próprio número. No máximo, um dos `defaultRegion` ou `defaultRegionColumn` pode ser fornecido.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

FormatCase classe

A `FormatCase` transformação altera cada string em uma coluna para o tipo de caso especificado.

Exemplo

```
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
```

Saída

A `FormatCase` transformação converterá os valores na coluna `cidade` em minúsculas com base no parâmetro `case_type="lower"`. O `df_output` resultante DataFrame conterá todas as colunas da `datasource1` original, mas com os valores da coluna DataFrame `cidade` em minúsculas.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_column, case_type)

A `FormatCase` transformação altera cada string em uma coluna para o tipo de caso especificado.

- `source_column`: o nome de uma coluna existente.
- `case_type`— Os tipos de casos compatíveis são`CAPITAL`,`LOWER`,`UPPER`,`SENTENCE`.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

FillWithMode classe

A `FillWithMode` transformação formata uma coluna de acordo com o formato do número de telefone especificado. Você também pode especificar a lógica de desempate, em que alguns dos valores são idênticos. Por exemplo, considere os seguintes valores: `1 2 2 3 3 4`

Um ModeType `MINIMUM` faz com que `FillWithMode` retorne 2 como o valor do modo. Se modeType for`MAXIMUM`, o modo será 3. Para`AVERAGE`, o modo é 2,5.

Exemplo

```
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
```

Saída

A saída do código fornecido 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|
+---------------+---------------+
```

A `FillWithMode` transformação do módulo `awsglue.data_quality` é aplicada ao `input_df`. DataFrame Ele substitui os valores `null` na `source_column_1` coluna pelo valor máximo (`mode_type="maximum"`) dos valores não nulos nessa coluna.

Nesse caso, o valor máximo na `source_column_1` coluna é `1055.123`. Portanto, os valores `null` em `source_column_1` são substituídos por `1055.123` na saída `df_output`. DataFrame

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_column, mode_type)

A `FillWithMode` transformação formata o caso das cadeias de caracteres em uma coluna.

- `source_column`: o nome de uma coluna existente.
- `mode_type`— Como resolver valores de tempo nos dados. Esse valor deve ser um de `MINIMUMNONE`,`AVERAGE`, ou`MAXIMUM`.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

FlagDuplicateRows classe

A `FlagDuplicateRows` transformação retorna uma nova coluna com um valor especificado em cada linha que indica se essa linha corresponde exatamente a uma linha anterior no conjunto de dados. Quando as correspondências são encontradas, elas são marcadas como duplicatas. A ocorrência inicial não é sinalizada porque não corresponde a uma linha anterior.

Exemplo

```
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
```

Saída

A saída será PySpark DataFrame com uma coluna adicional `flag_row` que indica se uma linha é duplicada ou não, com base na `source_column_1` coluna. O `df_output` DataFrame resultante conterá as seguintes linhas:

```
```
+---------------+---------------+--------+
|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
+---------------+---------------+--------+
```
```

A `flag_row` coluna indica se uma linha é duplicada ou não. A `true_string` é definida como “True” e a `false_string` está definida como “False”. O `target_index` é definido como 1, o que significa que a `flag_row` coluna será inserida na segunda posição (índice 1) na saída. 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=Nenhum)

A `FlagDuplicateRows` transformação retorna uma nova coluna com um valor especificado em cada linha que indica se essa linha corresponde exatamente a uma linha anterior no conjunto de dados. Quando as correspondências são encontradas, elas são marcadas como duplicatas. A ocorrência inicial não é sinalizada porque não corresponde a uma linha anterior.

- `true_string`— Valor a ser inserido se a linha corresponder a uma linha anterior.
- `false_string`— Valor a ser inserido se a linha for exclusiva.
- `target_column`— Nome da nova coluna inserida no conjunto de dados.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

RemoveDuplicates classe

A `RemoveDuplicates` transformação exclui uma linha inteira, se um valor duplicado for encontrado em uma coluna de origem selecionada.

Exemplo

```
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
```

Saída

A saída será PySpark DataFrame com as duplicatas removidas com base na `source_column_1` coluna. O `df_output` DataFrame resultante conterá as seguintes linhas:

```
```
+---------------+---------------+
|source_column_1|source_column_2|
+---------------+---------------+
105.111	13.12
13.12	13.12
null	13.12
+---------------+---------------+
```
```

Observe que as linhas com `source_column_1` valores de `13,12` e `null` aparecem somente uma vez na saída DataFrame, pois as duplicatas foram removidas com base na coluna. `source_column_1`

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_column)

A `RemoveDuplicates` transformação exclui uma linha inteira, se um valor duplicado for encontrado em uma coluna de origem selecionada.

- `source_column`: o nome de uma coluna existente.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

MonthName classe

A `MonthName` transformação cria uma nova coluna contendo o nome do mês, a partir de uma string que representa uma data.

Exemplo

```
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
```

Saída

A saída 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
+------------+------------+
```
```

A `MonthName` transformação usa a `source_column` como `"column_1"` e a `target_column` como `"target_column"`. Ele tenta extrair o nome do mês das sequências de data/hora na coluna `"column_1"` e o coloca na coluna `"target_column"`. Se a string de data/hora estiver em um formato não reconhecido ou não puder ser analisada, o valor `"target_column"` será definido como `null`.

A transformação extrai com sucesso o nome do mês de vários formatos de data/hora, como “20-12-2018", “20/12/2018", “02/02/2009", “2013-02-21 06:35:45.658 505" e “02 de agosto de 2009".

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform

- describeErrors
- describe

__call__ (spark_context, data_frame, target_column, source_column=Nenhum, valor=Nenhum)

A `MonthName` transformação cria uma nova coluna contendo o nome do mês, a partir de uma string que representa uma data.

- `source_column`: o nome de uma coluna existente.
- `value`— Uma sequência de caracteres para avaliar..
- `target_column`— Um nome para a coluna recém-criada.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

IsEven classe

A `IsEven` transformação retorna um valor booleano em uma nova coluna que indica se a coluna ou o valor de origem é par. Se a coluna ou o valor de origem for decimal, o resultado será falso.

Exemplo

```
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
```

Saída

A saída será:

```
```
+------------+------------+
|source_column|target_column|
+------------+------------+
```
```

```
| 5| Not even|
| 0| Even|
| -1| Not even|
| 2| Even|
| null| null|
+------------+------------+
```

A `IsEven` transformação usa a `source_column` como “source_column” e a `target_column` como “target_column”. Ele verifica se o valor na `"source_column"` é par ou não. Se o valor for par, ele define o valor `"target_column"` como `true_string` “Even”. Se o valor for ímpar, ele define o valor `"target_column"` como `false_string` “Nem mesmo”. Se o valor `"source_column"` for `null`, o valor `"target_column"` será definido como `null`.

A transformação identifica corretamente os números pares (0 e 2) e define o valor `"target_column"` como “Par”. Para números ímpares (5 e -1), ele define o valor `"target_column"` como “Nem par”. Para o valor `null` em `"source_column"`, o valor `"target_column"` é definido como `null`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, target_column, source_column=Nenhum, true_string=default_true_string, false_string=default_false_string, value=none)

A `IsEven` transformação retorna um valor booleano em uma nova coluna que indica se a coluna ou o valor de origem é par. Se a coluna ou o valor de origem for decimal, o resultado será falso.

- `source_column`: o nome de uma coluna existente.
- `target_column`— O nome da nova coluna a ser criada.

- `true_string`— Uma string que indica se o valor é par.
- `false_string`— Uma string que indica se o valor não é par.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

CryptographicHash classe

A `CryptographicHash` transformação aplica um algoritmo aos valores de hash na coluna.

Exemplo

```
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
```

Saída

A saída 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... |
+---+------------+------------------+------------------+
```

A transformação calcula os hashes criptográficos dos valores nas colunas `id` e `phone` usando o algoritmo e a chave secreta especificados e codifica os hashes no formato Base64. O `df_output` resultante DataFrame contém todas as colunas do `input_df` original, mais as colunas `id_hashed` e `phone_hashed` adicionais com os DataFrame hashes computados.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_columns, secret_id, algoritm=Nenhum, secret_version=Nenhum, create_secret_if_missing=false, output_format=Nenhum, entity_type_filter=Nenhum)

A `CryptographicHash` transformação aplica um algoritmo aos valores de hash na coluna.

- `source_columns`— Uma matriz de colunas existentes.
- `secret_id`— O ARN da chave secreta do Secrets Manager. A chave usada no algoritmo de prefixo do código de autenticação de mensagem baseado em hash (HMAC) para fazer o hash das colunas de origem.
- `secret_version`: opcional. O padrão é a versão secreta mais recente.

- `entity_type_filter`— Matriz opcional de tipos de entidades. Pode ser usado para criptografar somente as PII detectadas na coluna de texto livre.
- `create_secret_if_missing`— Booleano opcional. Se verdadeiro, tentará criar o segredo em nome do chamador.
- `algorithm`— O algoritmo usado para fazer o hash de seus dados. Valores de enumeração válidos: MD5, SHA1, SHA256, SHA512, HMAC_MD5, HMAC_SHA1, HMAC_SHA256, HMAC_SHA512.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe de decodificação

A `Decrypt` transformação é decifrada dentro do Glue. AWS Seus dados também podem ser descriptografados fora do AWS Glue com o AWS SDK de criptografia. Se o ARN da chave

KMS fornecida não corresponder ao que foi usado para criptografar a coluna, a operação de descriptografia falhará.

Exemplo

```
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
```

Saída

A saída será PySpark DataFrame com a coluna `id` original e a coluna `phone` descriptografada:

```
```
+---+------------+
| id| phone|
+---+------------+
1	1234560000
2	1234560001
3	1234560002
4	1234560003
5	1234560004
6	1234560005
7	1234560006
8	1234560007
9	1234560008
10	1234560009
+---+------------+
```
```

A `Encrypt` transformação usa `source_columns` como `["phone"]` e `kms_key_arn` como o valor da variável de ambiente `$ {KMS} `. A transformação criptografa os valores na coluna `telefone` usando a chave KMS especificada. O DataFrame `df_encrypt` criptografado é então passado para a transformação do módulo `awsglue.pii`. `Decrypt` Ele usa `source_columns` como `["phone"]` e `kms_key_arn` como o valor da variável de ambiente `$ {KMS} `. A transformação descriptografa os valores criptografados na coluna `telefone` usando a mesma chave KMS. O `df_decrypt` resultante DataFrame contém a coluna `id` original e a coluna `phone` descriptografada.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn

- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_columns, kms_key_arn)

A `Decrypt` transformação é decifrada dentro do Glue. AWS Seus dados também podem ser descriptografados fora do AWS Glue com o AWS SDK de criptografia. Se o ARN da chave KMS fornecida não corresponder ao que foi usado para criptografar a coluna, a operação de descriptografia falhará.

- `source_columns`— Uma matriz de colunas existentes.
- `kms_key_arn`— O ARN da chave do AWS Key Management Service a ser usada para descriptografar as colunas de origem.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

Classe de criptografia

A `Encrypt` transformação criptografa as colunas de origem usando a AWS chave do Serviço de Gerenciamento de Chaves. A `Encrypt` transformação pode criptografar até 128 MiB por célula. Ele tentará preservar o formato na decodificação. Para preservar o tipo de dados, os metadados do tipo de dados devem ser serializados para menos de 1 KB. Caso contrário, você deverá definir o `preserve_data_type` parâmetro como false. Os metadados do tipo de dados serão armazenados em texto simples no contexto de criptografia.

Exemplo

```
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
```

Saída

A saída será uma PySpark DataFrame com a coluna `id` original e uma coluna adicional contendo os valores criptografados da coluna `phone`.

```
```
+---+------------+------------------------+
| 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
+---+------------+------------------------+
```
```

A `Encrypt` transformação usa `source_columns` como `["phone"]` e `kms_key_arn` como o valor da variável de ambiente `$ {KMS} `. A transformação criptografa os valores na coluna `telefone` usando a chave KMS especificada. O `df_encrypt` resultante DataFrame contém a coluna `id` original, a coluna `phone` original e uma coluna adicional chamada `phone_encrypted` contendo os valores criptografados da coluna `phone`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform

- describeErrors
- describe

__call__ (spark_context, data_frame, source_columns, kms_key_arn, entity_type_filter=Nenhum, preserve_data_type=Nenhum)

A `Encrypt` transformação criptografa as colunas de origem usando a AWS chave do Serviço de Gerenciamento de Chaves.

- `source_columns`— Uma matriz de colunas existentes.
- `kms_key_arn`— O ARN da chave do AWS Key Management Service a ser usada para criptografar as colunas de origem.
- `entity_type_filter`— Matriz opcional de tipos de entidades. Pode ser usado para criptografar somente as PII detectadas na coluna de texto livre.
- `preserve_data_type`— Booleano opcional. O valor padrão é verdadeiro. Se for falso, o tipo de dados não será armazenado.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

IntToIp classe

A `IntToIp` transformação converte o valor inteiro da coluna de origem ou outro valor no valor IPv4 correspondente na coluna de destino e retorna o resultado em uma nova coluna.

Exemplo

```
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
```

Saída

A saída 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
+---------------+---------------+
```
```

A `IntToIp.apply` transformação usa a `source_column` como `"source_column_int"` e a `target_column` como `"target_column"` e converte os valores inteiros na coluna `source_column_int` em sua representação de endereço IPv4 correspondente e armazena o resultado na coluna `target_column`.

Para valores inteiros válidos dentro do intervalo de endereços IPv4 (0 a 4294967295), a transformação os converte com êxito em sua representação de endereço IPv4 (por exemplo, 192.0.0.1, 0.0.0.0, 10.0.0.10, 255.255.255.255).

Para valores inteiros fora do intervalo válido (por exemplo, 4294967296, -1), o valor `target_column` é definido como `null`. Para valores `null` na coluna `source_column_int`, o valor `target_column` também é definido como `null`.

Métodos

- __call__

- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, target_column, source_column=Nenhum, valor=Nenhum)

A `IntToIp` transformação converte o valor inteiro da coluna de origem ou outro valor no valor IPv4 correspondente na coluna de destino e retorna o resultado em uma nova coluna.

- `sourceColumn`: o nome de uma coluna existente.
- `value`— Uma sequência de caracteres para avaliar.
- `targetColumn`— O nome da nova coluna a ser criada.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

IpToInt classe

A `IpToInt` transformação converte o valor do Protocolo de Internet versão 4 (IPv4) da coluna de origem ou outro valor no valor inteiro correspondente na coluna de destino e retorna o resultado em uma nova coluna.

Exemplo

Para o AWS Glue 4.0 e versões posteriores, crie ou atualize argumentos de trabalho com `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()
```

Saída

A saída 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
+----------------+---------------+
```
```

A `IpToInt` transformação usa a `source_column` como `"source_column_ip"` e a `target_column` como `"target_column"` e converte as cadeias de endereço IPv4 válidas na coluna `source_column_ip` em sua representação inteira correspondente de 32 bits e armazena o resultado na coluna `target_column`.

Para cadeias de caracteres de endereço IPv4 válidas (por exemplo, “192.0.0.1", “10.10.10", “1.2.3.4"), a transformação as converte com êxito em sua representação inteira (por exemplo, 3221225473, 168427722, 16909060). Para strings que não são endereços IPv4 válidos (por exemplo, URLs, strings não IP como “abc”, formatos IP inválidos como “abc.abc.abc.abc”), o valor `target_column` é definido como `null`. Para valores `null` na coluna `source_column_ip`, o valor `target_column` também é definido como `null`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, target_column, source_column=Nenhum, valor=Nenhum)

A `IpToInt` transformação converte o valor do Protocolo de Internet versão 4 (IPv4) da coluna de origem ou outro valor no valor inteiro correspondente na coluna de destino e retorna o resultado em uma nova coluna.

- `sourceColumn`: o nome de uma coluna existente.
- `value`— Uma sequência de caracteres para avaliar.
- `targetColumn`— O nome da nova coluna a ser criada.

apply(cls, *args, **kwargs)

Herdado de `GlueTransform` apply.

name(cls)

Herdado de `GlueTransform` name.

describeArgs(cls)

Herdado de `GlueTransform` describeArgs.

describeReturn(cls)

Herdado de `GlueTransform` describeReturn.

describeTransform(cls)

Herdado de `GlueTransform` describeTransform.

describeErrors(cls)

Herdado de `GlueTransform` describeErrors.

describe(cls)

Herdado de `GlueTransform` describe.

A integração de dados se transforma

Para o AWS Glue 4.0 e versões posteriores, crie ou atualize argumentos de trabalho com`key: --enable-glue-di-transforms, value: true`.

Exemplo de script de trabalho:

```
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
```

Exemplo de sessões usando notebooks

```
%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
```

Exemplo de sessões usando AWS CLI

```
aws glue create-session --default-arguments "--enable-glue-di-transforms=true"
```

A DI transforma:

- FlagDuplicatesInColumn classe
- FormatPhoneNumber classe
- FormatCase classe
- FillWithMode classe
- FlagDuplicateRows classe
- RemoveDuplicates classe
- MonthName classe
- IsEven classe
- CryptographicHash classe
- Classe de decodificação
- Classe de criptografia
- IntToIp classe
- IpToInt classe

Maven: agrupe o plug-in com seus aplicativos Spark

Você pode agrupar a dependência de transformações com seus aplicativos Spark e distribuições Spark (versão 3.3) adicionando a dependência de plug-in em seu `pom.xml` Maven enquanto desenvolve seus aplicativos Spark localmente.

```
<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>
```

Como alternativa, você pode baixar os binários diretamente dos artefatos do AWS Glue Maven e incluí-los em seu aplicativo Spark da seguinte maneira.

```
#!/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/
```

Programando scripts de ETL do AWS Glue no Scala

É possível encontrar exemplos de código Scala e utilitários para o AWS Glue no Repositório de exemplos do AWS Glue no site do GitHub.

O AWS Glue oferece suporte a uma extensão do dialeto PySpark Scala para criação de scripts de trabalhos de extração, transformação e carregamento (ETL). As seções a seguir descrevem como usar a biblioteca Scala do AWS Glue e a API do AWS Glue em scripts de ETL e fornecem documentação de referência para a biblioteca.

Sumário

Usar o Scala para programar scripts de ETL do AWS Glue

Você pode gerar automaticamente um programa Scala de extração, transformação e carga (ETL) usando o console do AWS Glue e modificá-lo conforme necessário antes de atribui-lo a um trabalho. Ou, você pode escrever seu próprio programa do zero. Para obter mais informações, consulte Configurando as propriedades da tarefa para tarefas do Spark no AWS Glue. Em seguida, o AWS Glue compila seu programa Scala no servidor antes de executar o trabalho associado.

Para ter certeza de que seu programa compila sem erros e é executado como esperado, é muito importante carregá-lo em um endpoint de desenvolvimento em um REPL (Read-Eval-Print Loop) ou em um caderno Jupyter e testá-lo antes de executá-lo em um trabalho. Como o processo de compilação ocorre no servidor, você não terá uma boa visibilidade dos problemas que ocorrem lá.

Testar um programa de ETL Scala em um caderno Jupyter em um endpoint de desenvolvimento

Para testar um programa Scala em um endpoint de desenvolvimento do AWS Glue, configure o endpoint de desenvolvimento conforme descrito em Adicionar um endpoint de desenvolvimento.

Em seguida, conecte-o a um caderno Jupyter que esteja em execução localmente na sua máquina ou remotamente em um servidor de cadernos do Amazon EC2. Para instalar uma versão local de um caderno Jupyter, siga as instruções em Tutorial: caderno Jupyter no JupyterLab.

A única diferença entre executar o código Scala e executar o código PySpark no seu notebook é que você deve iniciar cada parágrafo no notebook com o seguinte:

```
%spark
```

Isso impede que o servidor do notebook seja padronizado para o tipo do PySpark do intérprete Spark.

Testar um programa de ETL Scala em um Scala REPL

Você pode testar um programa Scala em um endpoint do desenvolvedor usando o Scala REPL do AWS Glue. Siga as instruções em Tutorial: usar um caderno do SageMaker, exceto no final do comando SSH para REPL, substitua `-t gluepyspark` por `-t glue-spark-shell`. Isso invoca o Scala REPL do AWS Glue.

Para fechar o REPL quando você concluir, digite `sys.exit`.

Exemplo de script do Scala: ETL de transmissão

Example

O script de exemplo a seguir se conecta ao Amazon Kinesis Data Streams, usa um esquema do Data Catalog para analisar um fluxo de dados, une o fluxo a um conjunto de dados estático no Amazon S3 e gera os resultados unidos no Amazon S3 em 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()
  }
}
```

APIs na biblioteca Scala no AWS Glue

O AWS Glue oferece suporte a uma extensão do dialeto PySpark Scala para trabalhos de extração, transformação e carregamento (ETL) em script. As seções a seguir descrevem as APIs na biblioteca Scala do AWS Glue.

com.amazonaws.services.glue

O pacote com.amazonaws.services.glue na biblioteca Scala do AWS Glue contém as seguintes APIs:

- ChoiceOption
- DataSink
- Característica do DataSource

- DynamicFrame
- DynamicRecord
- GlueContext
- MappingSpec
- ResolveSpec

com.amazonaws.services.glue.ml

O pacote com.amazonaws.services.glue.ml na biblioteca de Scala do AWS Glue contém as seguintes APIs:

- FillMissingValues
- FindIncrementalMatches
- FindMatches

com.amazonaws.services.glue.dq

O pacote com.amazonaws.services.glue.ml na biblioteca Sacal da AWS Gluecontém as seguintes APIs:

- EvaluateDataQuality

com.amazonaws.services.glue.types

O pacote com.amazonaws.services.glue.types na biblioteca Scala do AWS Glue contém as seguintes APIs:

- ArrayNode
- BinaryNode
- BooleanNode
- ByteNode
- DateNode
- DecimalNode
- DoubleNode
- DynamicNode

- FloatNode
- IntegerNode
- LongNode
- MapLikeNode
- MapNode
- NullNode
- ObjectNode
- ScalarNode
- ShortNode
- StringNode
- TimestampNode

com.amazonaws.services.glue.util

O pacote com.amazonaws.services.glue.util na biblioteca Scala do AWS Glue contém as seguintes APIs:

- GlueArgParser
- Trabalho

APIs ChoiceOption em Scala no AWS Glue

Tópicos

- Característica ChoiceOption
- Objeto ChoiceOption
- Classe do caso ChoiceOptionWithResolver
- Classe do caso MatchCatalogSchemaChoiceOption

Pacote: com.amazonaws.services.glue

Característica ChoiceOption

```
trait ChoiceOption extends Serializable
```

Objeto ChoiceOption

ChoiceOption

```
object ChoiceOption
```

Uma estratégia geral para determinar a opção aplicável a todos os nós `ChoiceType` em um `DynamicFrame`.

- `val CAST`
- `val MAKE_COLS`
- `val MAKE_STRUCT`
- `val MATCH_CATALOG`
- `val PROJECT`

Def apply

```
def apply(choice: String): ChoiceOption
```

Classe do caso ChoiceOptionWithResolver

```
case class ChoiceOptionWithResolver(name: String, choiceResolver: ChoiceResolver)
 extends ChoiceOption {}
```

Classe do caso MatchCatalogSchemaChoiceOption

```
case class MatchCatalogSchemaChoiceOption() extends ChoiceOption {}
```

Classe abstrata DataSink

Tópicos

Pacote: com.amazonaws.services.glue

```
abstract class DataSink
```

O gravador analógico a um `DataSource`. O `DataSink` encapsula um destino e um formato no qual um `DynamicFrame` pode ser gravado.

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 do AWS Glue

Pacote: com.amazonaws.services.glue

Uma interface de alto nível para produzir um `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
}
```

APIs DynamicFrame em Scala do AWS Glue

Pacote: com.amazonaws.services.glue

Sumário

- 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

- O objeto DynamicFrame
  - Def apply

- Def emptyDynamicFrame
- Def fromPythonRDD
- Def ignoreErrors
- Def inlineErrors
- Def newFrameWithErrors

Classe DynamicFrame em Scala do AWS Glue

Pacote: 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 = {} )
```

Um `DynamicFrame` é uma coleção distribuída de objetos autodescritivos DynamicRecord.

`DynamicFrames` são projetados para fornecer um modelo de dados flexível para operações de ETL (extração, transformação e carregamento). Eles não exigem um esquema para criar e podem ser usados para ler e transformar dados com valores e tipos confusos ou inconsistentes. Um esquema pode ser calculado sob demanda para as operações que precisam de um.

`DynamicFrame` fornece uma variedade de transformações para limpeza de dados e ETL. Eles também suportam conversão de e para os DataFrames SparkSQL para integração com o código existente e as muitas operações de análise que os DataFrames fornecem.

Os seguintes parâmetros são compartilhados em muitas das transformações do AWS Glue que constroem `DynamicFrames`:

- `transformationContext`: o identificador para esse `DynamicFrame`. O `transformationContext` é usado como uma chave para o estado do marcador de trabalho que é mantido nas execuções.

- `callSite`: fornece informações de contexto para o relatório de erros. Esses valores são definidos automaticamente quando chamados do Python.
- `stageThreshold` — O número máximo de registros de erros permitido a partir da computação desse `DynamicFrame` antes de lançar uma exceção, excluindo os registros presentes no `DynamicFrame` anterior.
- `totalThreshold` – O número máximo de registros de erros totais antes que uma exceção seja lançada, incluindo os de quadros anteriores.

Val errorsCount

```
val errorsCount
```

O número de registros de erros neste `DynamicFrame`. Isso inclui erros de operações 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` — Uma sequência de mapeamentos para construir um novo `DynamicFrame`.
- `caseSensitive`: informa se deve ou não tratar as colunas de origem como colunas que diferenciam letras maiúsculas de minúsculas. Definir essa opção como false pode ajudar na integração com armazenamentos que não diferenciam maiúsculas de minúsculas, como o AWS Glue Data Catalog.

Seleciona, projeta e converte colunas com base em uma sequência de mapeamentos.

Cada mapeamento é composto por uma coluna e um tipo de origem e uma coluna e um tipo de destino. Mapeamentos podem ser especificados como um objeto 4-tuple (`source_path`, `source_type`, `target_path`, `target_type`) or a MappingSpec que contém a mesma informação.

Além de usar mapeamentos para projeções simples e da conversão, você pode usar os mapeamentos para aninhar ou desaninhar campos, separando os componentes do caminho com '`.`' (ponto).

Por exemplo, digamos que você tem um `DynamicFrame` com o seguinte esquema.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: struct
  |    |-- state: string
  |    |-- zip: int
}}}
```

Você pode fazer a seguinte chamada para desaninhar os campos `state` e `zip`.

```
{{{
  df.applyMapping(
   Seq(("name", "string", "name", "string"),
       ("age", "int", "age", "int"),
       ("address.state", "string", "state", "string"),
       ("address.zip", "int", "zip", "int")))
}}}
```

O esquema resultante será o seguinte.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- state: string
  |-- zip: int
}}}
```

Você também pode usar `applyMapping` para aninhar novamente as colunas. Por exemplo, o seguinte inverte a transformação anterior e cria uma estrutura chamada `address` no destino.

```
{{{
  df.applyMapping(
   Seq(("name", "string", "name", "string"),
       ("age", "int", "age", "int"),
```

```
        ("state", "string", "address.state", "string"),
        ("zip", "int", "address.zip", "int")))
 }}}
```

Os nomes de campo que contêm caracteres '`.`' (período) pode ser cotado usando acentos graves (``).

> **Note**
>
> Atualmente, você não pode usar o método `applyMapping` para mapear colunas que estão aninhadas sob arrays.

Def assertErrorThreshold

```
def assertErrorThreshold : Unit
```

Uma ação que força a computação e verifica se o número de registros de erros está abaixo de `stageThreshold` e de `totalThreshold`. Lança uma exceção se a condição falhar.

Def count

```
lazy
def count
```

Retorna o número de elementos neste `DynamicFrame`.

Def dropField

```
def dropField( path : String,
               transformationContext : String = "",
               callSite : CallSite = CallSite("Not provided", ""),
               stageThreshold : Long = 0,
               totalThreshold : Long = 0
             ) : DynamicFrame
```

Retorna um novo `DynamicFrame` com a coluna especificada removida.

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
```

Retorna um novo `DynamicFrame` com as colunas especificadas removidas.

Esse método pode ser usado para excluir colunas aninhadas, incluindo aquelas dentro de matrizes, mas não pode ser usado para descartar elementos de matriz específicos.

Def dropNulls

```
def dropNulls( transformationContext : String = "",
               callSite : CallSite = CallSite("Not provided", ""),
               stageThreshold : Long = 0,
               totalThreshold : Long = 0 )
```

Retorna um novo `DynamicFrame` com todas as colunas nulas removidas.

> **Note**
>
> Isso remove apenas colunas do tipo `NullType`. Valores nulos individuais em outras colunas não são removidos ou modificados.

Def errorsAsDynamicFrame

```
def errorsAsDynamicFrame
```

Retorna um novo `DynamicFrame` contendo os registros de erro deste `DynamicFrame`.

Def filter

```
def filter( f : DynamicRecord => Boolean,
            errorMsg : String = "",
            transformationContext : String = "",
            callSite : CallSite = CallSite("Not provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
          ) : DynamicFrame
```

Cria um novo `DynamicFrame` que contém apenas os registros para os quais a função "`f`" retorna `true`. A função de filtro "`f`" não deve alterar o registro de entrada.

Def getName

```
def getName : String
```

Retorna o nome deste `DynamicFrame`.

Def getNumPartitions

```
def getNumPartitions
```

Retorna o número de partições neste `DynamicFrame`.

Def getSchemaIfComputed

```
def getSchemaIfComputed : Option[Schema]
```

Retorna o esquema, se ele já foi calculado. Não verifica os dados se o esquema ainda não tiver sido calculado.

Def isSchemaComputed

```
def isSchemaComputed : Boolean
```

Retorna `true` se o esquema foi calculado para este `DynamicFrame` ou `false`, se não. Se esse método retorna falso, chamar o método `schema` exige outra passagem pelos registros nesse `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` – As colunas nesse `DynamicFrame` para usar na junção.
- `keys2`: as colunas no `frame2` a serem usadas na junção. Deve ter o mesmo tamanho que `keys1`.
- `frame2` — O `DynamicFrame` para usar na junção.

Retorna o resultado da execução de uma junção equivalente com `frame2` usando as chaves especificadas.

Def map

```
def map( f : DynamicRecord => DynamicRecord,
         errorMsg : String = "",
         transformationContext : String = "",
         callSite : CallSite = CallSite("Not provided", ""),
         stageThreshold : Long = 0,
         totalThreshold : Long = 0
       ) : DynamicFrame
```

Retorna um novo `DynamicFrame` criado aplicando a função especificada "f" para cada registro nesse `DynamicFrame`.

Este método copia cada registro antes de aplicar a função especificada, portanto, é seguro alterar os registros. Se a função de mapeamento lançar uma exceção em um determinado registro, esse registro é marcado como um erro e o rastreamento de pilha é salvo como uma coluna no registro de erros.

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`: o `DynamicFrame` de preparação para mesclar.

- `primaryKeys`: a lista de campos de chave primária para corresponder aos registros da fonte e `DynamicFrames` de preparação.
- `transformationContext`: uma string exclusiva usada para recuperar os metadados sobre a transformação atual (opcional).
- `options`: uma string de pares nome-valor JSON que fornecem informações adicionais para essa transformação.
- `callSite`: usado para fornecer informações de contexto para o relatório de erros.
- `stageThreshold`: uma `Long`. O número de erros na transformação para a qual o processamento precisa apresentar falhas.
- `totalThreshold`: uma `Long`. O número total de erros nessa transformação para os quais o processamento precisa apresentar falhas.

Mescla esse `DynamicFrame` com uma preparação `DynamicFrame` de acordo com as chaves primárias especificadas para identificar registros. Registros duplicados (com as mesmas chaves primárias) não são eliminados. Se não houver nenhum registro correspondente no quadro de preparação, todos os registros (incluindo os duplicados) serão retidos da origem. Se o quadro de preparação tiver registros correspondentes, os do quadro de preparação substituirão os da origem no AWS Glue.

O `DynamicFrame` retornado contém registro A nos seguintes casos:

1. Se `A` existir no quadro de origem e no quadro de preparação, o `A` do quadro de preparação será retornado.
2. Se `A` estiver na tabela de origem e `A.primaryKeys` não estiver no `stagingDynamicFrame` (isso significa que `A` não é atualizado na tabela de preparação).

O quadro de origem e o quadro de preparação não precisam ter o mesmo esquema.

Example

```
val mergedFrame: DynamicFrame = srcFrame.mergeDynamicFrames(stageFrame, Seq("id1",
 "id2"))
```

Def printSchema

```
def printSchema : Unit
```

Imprime o esquema deste `DynamicFrame` para `stdout` em um formato legível.

Def recomputeSchema

```
def recomputeSchema : Schema
```

Força um recálculo do esquema. Isso requer uma verificação nos dados, mas pode "restringir" o esquema se houver alguns campos no esquema atual que não estejam presentes nos dados.

Retorna o esquema recalculado.

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`: o nome a ser usado para o `DynamicFrame` base na saída. `DynamicFrames` criados pelo deslocamento de matrizes começam com isso como um prefixo.
- `stagingPath`: o caminho do Amazon Simple Storage Service (Amazon S3) para gravação de dados intermediários.
- `options`: opções e a configuração da função Relationalize. Não utilizado no momento.

Nivela todas as estruturas aninhadas e matrizes dinâmicas em tabelas separadas.

Você pode usar essa operação para preparar dados extremamente aninhados para adição em um banco de dados relacional. As estruturas aninhadas são niveladas da mesma maneira que a transformação Unnest. Além disso, as matrizes são dinamizadas em tabelas separadas, com cada elemento da matriz se tornando uma linha. Por exemplo, digamos que você tenha um `DynamicFrame` com os seguintes dados.

```
{"name": "Nancy", "age": 47, "friends": ["Fred", "Lakshmi"]}
{"name": "Stephanie", "age": 28, "friends": ["Yao", "Phil", "Alvin"]}
```

```
 {"name": "Nathan", "age": 54, "friends": ["Nicolai", "Karen"]}
```

Execute o código a seguir.

```
{{{
  df.relationalize("people", "s3:/my_bucket/my_path", JsonOptions.empty)
}}}
```

Isso produz duas tabelas. A primeira tabela é chamada "pessoas" e contém o seguinte.

```
{{{
  {"name": "Nancy", "age": 47, "friends": 1}
  {"name": "Stephanie", "age": 28, "friends": 2}
  {"name": "Nathan", "age": 54, "friends": 3)
}}}
```

Aqui, a matriz de amigos foi substituída por uma chave de junção gerada automaticamente. Uma tabela separada chamada `people.friends` é criada com o seguinte conteúdo.

```
{{{
  {"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"}
}}}
```

Nessa tabela, '`id`' é uma chave de junção que identifica de qual registro o elemento da matriz é proveniente, '`index`' refere-se à posição na matriz original e '`val`' é a entrada real da matriz.

O método `relationalize` retorna a sequência de `DynamicFrame` criada aplicando esse processo recursivamente a todas as matrizes.

> **Note**
>
> A biblioteca do AWS Glue gera automaticamente chaves de junção para novas tabelas. Para garantir que as chaves de junção sejam exclusivas nas execuções do trabalho, você deve ativar as marcações de trabalho.

Def renameField

```
def renameField( oldName : String,
                 newName : String,
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : DynamicFrame
```

- `oldName` – O nome original da coluna.
- `newName` – O novo nome da coluna.

Retorna um novo `DynamicFrame` com campo especificado renomeado.

Você pode usar esse método para renomear campos aninhados. Por exemplo, o código a seguir renomearia `state` como `state_code` na estrutura de endereço.

```
{{{
  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
```

Retorna um novo `DynamicFrame` com partições `numPartitions`.

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` — Uma ação a ser aplicada a todas as colunas `ChoiceType` não listadas na sequência de especificações.
- `database`: o banco de dados do Data Catalog a ser usado com a ação `match_catalog`.
- `tableName`: a tabela do Data Catalog a ser usada com a ação `match_catalog`.

Retorna um novo `DynamicFrame` substituindo um ou mais `ChoiceType` por um tipo mais específico.

Há duas maneiras de usar `resolveChoice`. A primeira é especificar uma sequência de colunas específicas e como resolvê-las. Elas são especificadas como tuplas compostas de pares (coluna, ação).

A seguir estão as ações possíveis:

- `cast:type` – Tenta converter todos os valores para o tipo especificado.
- `make_cols` – Converte cada tipo distinto em uma coluna com o nome `columnName_type`.
- `make_struct` – Converte uma coluna em uma estrutura com chaves para cada tipo distinto.
- `project:type` — Mantém apenas valores do tipo especificado.

O outro modo para `resolveChoice` é especificar uma única resolução para todos os `ChoiceTypes`. Você pode usar isso em casos em que a lista completa de `ChoiceTypes` for desconhecida antes da execução. Além das ações listadas anteriormente, esse modo também aceita a seguinte ação:

- `match_catalogChoiceType` – Tenta converter cada para o tipo correspondente na tabela de catálogo especificada.

Exemplos:

Corrija a coluna `user.id` convertendo para um inteiro e faça o campo `address` reter apenas as estruturas.

```
{{{
  df.resolveChoice(specs = Seq(("user.id", "cast:int"), ("address", "project:struct")))
}}}
```

Resolva todos os `ChoiceTypes` convertendo cada opção em uma coluna separada.

```
{{{
  df.resolveChoice(choiceOption = Some(ChoiceOption("make_cols")))
}}}
```

Resolva todos os `ChoiceTypes` convertendo para os tipos na tabela de catálogo especificada.

```
{{{
  df.resolveChoice(choiceOption = Some(ChoiceOption("match_catalog")),
                   database = Some("my_database"),
                   tableName = Some("my_table"))
}}}
```

Def schema

```
def schema : Schema
```

Retorna o esquema deste `DynamicFrame`.

O esquema retornado é garantido para conter todos os domínios que estão presentes nesse registro `DynamicFrame`. Mas em um pequeno número de casos, ele também pode conter campos adicionais. Você pode usar o método Unnest para "restringir" o esquema com base nos registros deste `DynamicFrame`.

Def selectField

```
def selectField( fieldName : String,
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : DynamicFrame
```

Retorna um único campo como um `DynamicFrame`.

Def selectFields

```
def selectFields( paths : Seq[String],
                  transformationContext : String = "",
                  callSite : CallSite = CallSite("Not provided", ""),
                  stageThreshold : Long = 0,
                  totalThreshold : Long = 0
                ) : DynamicFrame
```

- `paths` — A sequência de nomes de coluna para selecionar.

Retorna um novo `DynamicFrame` contendo as colunas especificadas.

> **Note**
>
> O método `selectFields` só pode ser usado para selecionar as colunas de nível superior. O método applyMapping pode ser usado para selecionar as colunas aninhadas.

Def show

```
def show( numRows : Int = 20 ) : Unit
```

- `numRows` – O número de linhas a serem impressas.

Imprime linhas desse `DynamicFrame` no formato JSON.

Def simplifyDDBJson

As exportações do DynamoDB feitas com o conector de exportação para DynamoDB do AWS Glue resultam em arquivos JSON de estruturas aninhadas específicas. Para obter mais informações, consulte Objetos de dados. `simplifyDDBJson` Simplifica as colunas aninhadas em um DynamicFrame desse tipo de dados e retorna um novo DynamicFrame simplificado. Se houver vários tipos ou um tipo de mapa contidos em um tipo de lista, os elementos na lista não serão simplificados. Esse método oferece suporte somente a dados no for mato JSON de exportação do DynamoDB. Considere o `unnest` para realizar alterações semelhantes em outros tipos de dados.

```
def simplifyDDBJson() : DynamicFrame
```

Esse método não aceita parâmetros.

Exemplo de entrada

Considere o seguinte esquema gerado por uma exportação do 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 exemplo

```
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()
  }

}
```

Exemplo de saída

A transformação `simplifyDDBJson` simplificará isso para:

```
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
```

Transformação de passagem que retorna os mesmos registros, mas grava um subconjunto de registros como um efeito secundário.

- `path`: o caminho no Amazon S3 no qual gravar a saída, no formato `s3://bucket//path`.
- `options` — Um mapa opcional `JsonOptions` que descreve o comportamento de amostragem.

Retorna um `DynamicFrame` que contém os mesmos registros deste.

Por padrão, grava 100 registros arbitrários no local especificado por `path`. Você pode personalizar esse comportamento usando o mapa `options`. Chaves válidas incluem o seguinte:

- `topk`: especifica o número total de registros gravados. O padrão é 100.
- `prob`: especifica a probabilidade (como um número decimal) de incluir um registro individual. O padrão é 1.

Por exemplo, a chamada a seguir seria um exemplo do conjunto de dados, selecionando cada registro com uma probabilidade de 20% e interrompendo após a gravação de 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` — Os caminhos a serem incluídos no primeiro `DynamicFrame`.

Retorna uma sequência de dois `DynamicFrames`. O primeiro `DynamicFrame` contém os caminhos especificados e o segundo, todas as outras colunas.

Exemplo

Este exemplo usa um DynamicFrame criado a partir da tabela `persons` no banco de dados `legislators` no catálogo de dados do Glue da AWS e divide o DynamicFrame em dois, com os campos especificados entrando no primeiro DynamicFrame e os campos restantes entrando em um segundo DynamicFrame. O exemplo então escolhe o primeiro DynamicFrame do 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 linhas com base em predicados que comparam colunas com constantes.

- `paths` — As colunas a serem usadas para comparação.
- `values` — Os valores constantes a serem usados para comparação.
- `operators` — Os operadores a serem usados para comparação.

Retorna uma sequência de dois `DynamicFrames`. O primeiro contém linhas para as quais o predicado é verdadeiro, e o segundo contém linhas para as quais ele é falso.

Os predicados são especificados usando três sequências: "`paths`" contém os nomes de coluna (possivelmente aninhados), "`values`" contém os valores constantes para comparar e "`operators`" contém os operadores a serem usados para comparação. Todas as três sequências devem ter o mesmo tamanho: o operador n é usado para comparar a coluna n com o valor n.

Cada operador deve ser um de "!=", "=", "<=", "<", ">=" ou ">".

Como exemplo, a chamada a seguir dividiria um `DynamicFrame` para que o primeiro quadro de saída contivesse registros de pessoas dos Estados Unidos com mais de 65 anos, e o segundo contivesse todos os outros registros.

```
{{{
  df.splitRows(Seq("age", "address.country"), Seq(65, "USA"), Seq("&gt;=", "="))
}}}
```

Def stageErrorsCount

```
def stageErrorsCount
```

Retorna o número de registros de erros criados durante o cálculo desse `DynamicFrame`. Isso exclui erros de operações anteriores que foram passadas para este `DynamicFrame` como entrada.

Def toDF

```
def toDF( specs : Seq[ResolveSpec] = Seq.empty[ResolveSpec] ) : DataFrame
```

Converte isso `DynamicFrame` para um Apache Spark SQL `DataFrame` com o mesmo esquema e registros.

> **Note**
>
> Como `DataFrames` não oferecem suporte para `ChoiceTypes`, esse método converte automaticamente colunas `ChoiceType` em `StructTypes`. Para obter mais informações e opções para opções de resolução, 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`: a coluna a ser analisada. Deve ser uma string ou binária.
- `format` – O formato a ser usado para a análise.
- `optionString` – Opções para passar para o formato, como o separador CSV.

Analisa uma coluna de string ou binária incorporada de acordo com o formato especificado. Colunas analisadas são aninhadas sob uma estrutura com o nome da coluna original.

Por exemplo, suponha que há um arquivo CSV com uma coluna JSON incorporada,

```
name, age, address
Sally, 36, {"state": "NE", "city": "Omaha"}
...
```

Após uma análise inicial, seria obtido um `DynamicFrame` com o seguinte esquema.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: string
}}}
```

É possível chamar `unbox` na coluna de endereço para analisar os componentes específicos.

```
{{{
  df.unbox("address", "json")
}}}
```

Isso nos dá um `DynamicFrame` com o seguinte 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
```

Retorna um novo `DynamicFrame` com todas as estruturas aninhadas niveladas. Os nomes são criados usando o caractere `'.'` (ponto).

Por exemplo, digamos que você tem um `DynamicFrame` com o seguinte esquema.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: struct
  |    |-- state: string
  |    |-- city: string
}}}
```

A chamada a seguir desaninha a estrutura do endereço.

```
{{{
  df.unnest()
}}}
```

O esquema resultante será o seguinte.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address.state: string
  |-- address.city: string
}}}
```

Esse método também desaninha estruturas aninhados dentro de matrizes. No entanto, por razões históricas, os nomes dos campos são prefixados com o nome do array de fechamento e "`.val`".

Def unnestDDBJson

```
unnestDDBJson(transformationContext : String = "",
            callSite : CallSite = CallSite("Not Provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0): DynamicFrame
```

Desaninha colunas aninhadas em um `DynamicFrame` que estão especificamente na estrutura JSON do DynamoDB e retorna um novo `DynamicFrame` não aninhado. Colunas que pertençam a uma matriz de tipos de estrutura não serão desaninhadas. Observe que esse é um tipo específico de transformação de desaninhamento que se comporta diferentemente da transformação `unnest` comum e requer que os dados já estejam na estrutura JSON do DynamoDB. Para mais informações, consulte JSON do DynamoDB.

Por exemplo, o esquema de uma leitura de uma exportação com a estrutura JSON do DynamoDB pode se parecer com o seguinte:

```
root
|-- Item: struct
|    |-- ColA: struct
|    |    |-- S: string
|    |-- ColB: struct
|    |    |-- S: string
|    |-- ColC: struct
```

```
|    |    |-- N: string
|    |-- ColD: struct
|    |    |-- L: array
|    |    |    |-- element: null
```

A transformação `unnestDDBJson()` converteria isso em:

```
root
|-- ColA: string
|-- ColB: string
|-- ColC: string
|-- ColD: array
|    |-- element: null
```

O exemplo de código a seguir mostra como usar o conector de exportação para DynamoDB do AWS Glue, invocar um desaninhamento de JSON do DynamoDB e imprimir o número de partições:

```
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`: uma função que retorna o esquema a ser usado. Especificado como uma função de parâmetro zero para adiar a computação potencialmente dispendiosa.

Define o esquema desse `DynamicFrame` com o valor especificado. Isso é usado principalmente de maneira interna para evitar a recomputação dispendiosa do esquema. O esquema transmitido deve conter todas as colunas presentes nos dados.

Def withName

```
def withName( name : String ) : DynamicFrame
```

- `name` — O novo nome a ser usado.

Retorna uma cópia desse `DynamicFrame` com um novo nome.

Def withTransformationContext

```
def withTransformationContext( ctx : String ) : DynamicFrame
```

Retorna uma cópia desse `DynamicFrame` com o contexto de transformação especificado.

O objeto DynamicFrame

Pacote: 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
```

Classe DynamicRecord em Scala do AWS Glue

Tópicos

Pacote: com.amazonaws.services.glue

```
class DynamicRecord extends Serializable with Writable with Cloneable
```

Um `DynamicRecord` é uma estrutura de dados autodescritiva que representa uma linha de conjunto de dados que está sendo processado. É autodescritiva, no sentido de que você pode obter o esquema da linha representada pelo `DynamicRecord`, inspecionando o próprio registro. Um `DynamicRecord` é semelhante a uma `Row` no Apache Spark

Def addField

```
def addField( path : String,
```

```
              dynamicNode : DynamicNode
            ) : Unit
```

Adiciona um DynamicNode ao caminho especificado.

- `path` – O caminho para o campo a ser adicionado.
- `dynamicNode` — O DynamicNode a ser adicionado no caminho especificado.

Def dropField

```
def dropField(path: String, underRename: Boolean = false): Option[DynamicNode]
```

Remove um DynamicNode do caminho especificado e retorna o nó removido se não houver uma matriz no caminho especificado.

- `path` — O caminho para o campo a ser removido.
- `underRenamedropField` – Verdadeiro se for chamado como parte de uma transformação de renomeação ou falso, caso contrário (falso por padrão).

Retorna um `scala.Option Option` (DynamicNode).

Def setError

```
def setError( error : Error )
```

Define esse registro como um registro de erro, conforme especificado pelo parâmetro `error`.

Retorna um `DynamicRecord`.

Def isError

```
def isError
```

Verifica se esse registro é um registro de erro.

Def getError

```
def getError
```

Obtém o `Error` se o registro for um registro de erro. Retorna `scala.Some` Some (Erro) se esse registro for um registro de erro ou `scala.None`, caso contrário.

Def clearError

```
def clearError
```

Defina `Error` como `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 esse registro para um novo `DynamicRecord` e o retorna.

Def schema

```
def schema
```

Obtém o `Schema` ao inspecionar o registro.

Def getRoot

```
def getRoot : ObjectNode
```

Obtém a raiz `ObjectNode` do registro.

Def toJson

```
def toJson : String
```

Obtém a string JSON para o registro.

Def getFieldNode

```
def getFieldNode( path : String ) : Option[DynamicNode]
```

Obtém o valor do campo no `path` especificado como uma opção de `DynamicNode`.

Retorna `scala.Some` Some (DynamicNode) se o campo existir ou `scala.None.None`, caso contrário.

Def getField

```
def getField( path : String ) : Option[Any]
```

Obtém o valor do campo no `path` especificado como uma opção de `DynamicNode`.

Retorna `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 o método para converter um Apache Spark SQL `Row` em um DynamicRecord.

- `row` – Um Spark SQL Row.
- `schema` – O Schema dessa linha.

Retorna um `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 GlueAPIs Scala GlueContext

Pacote: com.amazonaws.services.glue

```
class GlueContext extends SQLContext(sc) (
           @transient val sc : SparkContext,
           val defaultSourcePartitioner : PartitioningStrategy )
```

`GlueContext` é o ponto de entrada para leitura e gravação de um DynamicFrame de e para o Amazon Simple Storage Service (Amazon S3), AWS Glue Data Catalog, JDBC e assim por diante.

Essa classe fornece funções de utilitário para criar objetos Característica do DataSource e DataSink que, por sua vez, podem ser usados para ler e gravar `DynamicFrames`.

`GlueContext` também pode ser usado para definir um número de partições de destino (o padrão é 20) no `DynamicFrame` se o número de partições criado da origem é menor do que um limite mínimo para as partições (o padrão é 10).

colunas def addIngestionTime

```
def addIngestionTimeColumns(
         df : DataFrame,
         timeGranularity : String = "") : dataFrame
```

Acrescenta colunas de tempo de ingestão, como `ingest_year`, `ingest_month`, `ingest_day`, `ingest_hour`, `ingest_minute`, para o `DataFrame` de entrada. Essa função é gerada automaticamente no script gerado pelo AWS Glue, quando você especifica uma tabela do catálogo de dados com o Amazon S3 como destino. Essa função atualiza automaticamente a partição com colunas de tempo de ingestão na tabela de saída. Isso permite que os dados de saída sejam particionados automaticamente no tempo de ingestão sem exigir colunas de tempo de ingestão explícitas nos dados de entrada.

- `dataFrame`: o `dataFrame` ao qual anexar as colunas de tempo de ingestão.
- `timeGranularity`: o detalhamento das colunas de tempo. Os valores válidos são “`day`”, “`hour`” e “`minute`”. Por exemplo, se “`hour`” é transmitido para a função, o `dataFrame` original terá as colunas de tempo “`ingest_year`”, “`ingest_month`”, “`ingest_day`” e “`ingest_hour`” anexadas.

Retorna o quadro de dados após anexar as colunas de detealhamento de tempo.

Exemplo:

```
glueContext.addIngestionTimeColumns(dataFrame, "hour")
```

definitivamente createDataFrame FromOptions

```
def createDataFrameFromOptions( connectionType : String,
                        connectionOptions : JsonOptions,
                        transformationContext : String = "",
```

```
                        format : String = null,
                        formatOptions : JsonOptions = JsonOptions.empty
                      ) : DataSource
```

Retorna um `DataFrame` criado com a conexão e o formato especificados. Use essa função somente com fontes de streaming do AWS Glue.

- `connectionType`: o tipo de conexão de transmissão. Os valores válidos são `kinesis` e `kafka`.
- `connectionOptions`: opções de conexão, a quais são diferentes para Kinesis e Kafka. Você pode encontrar a lista de todas as opções de conexão para cada origem dos dados de transmissão em Tipos e opções de conexão para ETL no AWS Glue para Spark. Observe as seguintes diferenças nas opções de conexão de transmissão:
  - As fontes de transmissão do Kinesis exigem `streamARN`, `startingPosition`, `inferSchema` e `classification`.
  - As fontes de transmissão do Kafka exigem `connectionName`, `topicName`, `startingOffsets`, `inferSchema` e `classification`.
- `transformationContext`: o contexto de transformação a ser usado (opcional).
- `format`: uma especificação de formato (opcional). É usado para uma conexão do Amazon S3 ou do AWS Glue com suporte para vários formatos. Para obter informações sobre os formatos compatíveis, consulte Opções de formato de dados para entradas e saídas no AWS Glue para Spark.
- `formatOptions`: as opções de formato para o formato especificado. Para obter informações sobre as opções de formato compatíveis, consulte Opções de formato de dados.

Exemplo para fonte de transmissão do Amazon Kinesis:

```
val data_frame_datasource0 =
glueContext.createDataFrameFromOptions(transformationContext = "datasource0",
 connectionType = "kinesis",
connectionOptions = JsonOptions("""{"streamName": "example_stream", "startingPosition":
 "TRIM_HORIZON", "inferSchema": "true", "classification": "json"}}""" ))
```

Exemplo para fonte de transmissão do 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)`

Aplica a `batch_function` transmitida para cada microlote lido a partir da fonte de transmissão.

- `frame`— O DataFrame que contém o microlote atual.
- `batch_function`: uma função que será aplicada para cada microlote.
- `options`: uma coleção de pares de chave-valor que contém informações sobre como processar microlotes. São necessárias as seguintes opções:
  - `windowSize`: a quantidade de tempo gasto no processamento de cada lote.
  - `checkpointLocation`: o local onde os pontos de verificação são armazenados para o trabalho de ETL de transmissão.
  - `batchMaxRetries`: o número máximo de novas tentativas deste lote em caso de falha. O valor padrão é 3. Essa opção só pode ser configurada para o Glue versão 2.0 e posterior.

Exemplo:

```
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
```

Cria um DataSink que grava em um local especificado de uma tabela definida no Data Catalog.

- `database`: o nome do banco de dados no Data Catalog.
- `tableName`: o nome da tabela no Data Catalog.
- `redshiftTmpDir`: o diretório de preparação temporário a ser usado com determinados coletores de dados. Definido como vazio por padrão.
- `transformationContext`: o contexto de transformação associado ao coletor a ser usado pelos marcadores de trabalho. Definido como vazio por padrão.
- `additionalOptions`: opções adicionais fornecidas ao AWS Glue.
- `catalogId`: o ID do catálogo (ID da conta) do Data Catalog que está sendo acessado. Quando nulo, o ID da conta do chamador padrão é usado.

Retorna o `DataSink`.

definitivamente getCatalogSource

```
def getCatalogSource( database : String,
```

```
                    tableName : String,
                    redshiftTmpDir : String = "",
                    transformationContext : String = ""
                    pushDownPredicate : String = " "
                    additionalOptions: JsonOptions = JsonOptions.empty,
                    catalogId: String = null
                  ) : DataSource
```

Cria um Característica do DataSource que lê dados em uma definição de tabela do Data Catalog.

- `database`: o nome do banco de dados no Data Catalog.
- `tableName`: o nome da tabela no Data Catalog.
- `redshiftTmpDir`: o diretório de preparação temporário a ser usado com determinados coletores de dados. Definido como vazio por padrão.
- `transformationContext`: o contexto de transformação associado ao coletor a ser usado pelos marcadores de trabalho. Definido como vazio por padrão.
- `pushDownPredicate`: filtra partições sem a necessidade de listar e ler todos os arquivos no seu conjunto de dados. Para ter mais informações, consulte Pré-filtragem usando a aplicação de predicados.
- `additionalOptions` – Uma coleção de pares nome-valor opcionais. As opções possíveis incluem as listadas em Tipos e opções de conexão para ETL no AWS Glue para Spark, exceto para `endpointUrl`, `streamName`, `bootstrap.servers`, `security.protocol`, `topicName`, `classification` e `delimiter`. Outra opção suportada é `catalogPartitionPredicate`:

  `catalogPartitionPredicate`: você pode transmitir uma expressão de catálogo para filtrar com base nas colunas de índice. Isso leva a filtragem para o lado do servidor. Para obter mais informações, consulte Índices de partição do AWS Glue. Observe que `push_down_predicate` e `catalogPartitionPredicate` usam sintaxes diferentes. O primeiro usa a sintaxe padrão do Spark SQL e o outro usa o analisador JSQL.
- `catalogId`: o ID do catálogo (ID da conta) do Data Catalog que está sendo acessado. Quando nulo, o ID da conta do chamador padrão é usado.

Retorna o `DataSource`.

Exemplo de fonte de transmissão

```
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
```

Cria um DataSink que grava em um banco de dados JDBC especificado em um objeto `Connection` no Data Catalog. O objeto `Connection` tem informações para se conectar a um depósito JDBC, incluindo URL, nome de usuário, senha, VPC, sub-rede e grupos de segurança.

- `catalogConnection`: o nome da conexão no Data Catalog que contém o URL do JDBC onde gravar.
- `options`: uma string de pares de nome-valor JSON que fornecem as informações adicionais necessárias para gravar em um datastore do JDBC. Isso inclui:
  - dbtable (obrigatório) — O nome da tabela JDBC. Para armazenamentos de dados JDBC que oferecem suporte a esquemas dentro de um banco de dados, especifique `schema.table-name`. Se um esquema não for fornecido, o esquema "público" padrão será usado. O exemplo a seguir mostra um parâmetro de opções que aponta para um esquema chamado `test` e uma tabela chamada `test_table` no banco de dados `test_db`.

    ```
    options = JsonOptions("""{"dbtable": "test.test_table", "database": "test_db"}""")
    ```

  - database (obrigatório) — O nome do banco de dados do JDBC.
  - Todas as opções adicionais transmitidas diretamente para gravador do JDBC de SparkSQL. Para obter mais informações, consulte Fonte de dados Redshift para Spark.
- `redshiftTmpDir`: um diretório de preparação temporário a ser usado com determinados coletores de dados. Definido como vazio por padrão.

- `transformationContext`: o contexto de transformação associado ao coletor a ser usado pelos marcadores de trabalho. Definido como vazio por padrão.
- `catalogId`: o ID do catálogo (ID da conta) do Data Catalog que está sendo acessado. Quando nulo, o ID da conta do chamador padrão é usado.

Código de exemplo:

```
getJDBCSink(catalogConnection = "my-connection-name", options =
 JsonOptions("""{"dbtable": "my-jdbc-table", "database": "my-jdbc-db"}"""),
 redshiftTmpDir = "", transformationContext = "datasink4")
```

Retorna o `DataSink`.

def getSink

```
def getSink( connectionType : String,
             connectionOptions : JsonOptions,
             transformationContext : String = ""
           ) : DataSink
```

Cria um DataSink que grava dados em um destino como o Amazon Simple Storage Service (Amazon S3), o JDBC ou o Glue Data Catalog, ou um stream de dados AWS do Apache Kafka ou do Amazon Kinesis.

- `connectionType` — O tipo da conexão. Consulte the section called “Parâmetros de conexão”.
- `connectionOptions` – uma string de pares de nome/valor JSON que fornecem as informações adicionais para estabelecer a conexão com o depósito de dados. Consulte the section called “Parâmetros de conexão”.
- `transformationContext`: o contexto de transformação associado ao coletor a ser usado pelos marcadores de trabalho. Definido como vazio por padrão.

Retorna o `DataSink`.

Formato def getSinkWith

```
def getSinkWithFormat( connectionType : String,
                       options : JsonOptions,
```

```
                        transformationContext : String = "",
                        format : String = null,
                        formatOptions : JsonOptions = JsonOptions.empty
                      ) : DataSink
```

Cria um DataSink que grava dados em um destino como Amazon S3, JDBC, Catálogo de Dados ou um fluxo de dados do Amazon Kinesis ou do Apache Kafka. Também define o formato dos dados que serão gravados no destino.

- `connectionType` — O tipo da conexão. Consulte the section called “Parâmetros de conexão”.
- `options` — Uma string de pares de nome/valor JSON que fornecem as informações adicionais para estabelecer a conexão com o depósito de dados. Consulte the section called “Parâmetros de conexão”.
- `transformationContext`: o contexto de transformação associado ao coletor a ser usado pelos marcadores de trabalho. Definido como vazio por padrão.
- `format` — O formato dos dados a serem gravados no destino.
- `formatOptions`: uma string de pares de nome-valor JSON que fornecem opções adicionais para a formatação de dados no destino. Consulte Opções de formato de dados.

Retorna o `DataSink`.

def getSource

```
def getSource( connectionType : String,
               connectionOptions : JsonOptions,
               transformationContext : String = ""
               pushDownPredicate
             ) : DataSource
```

Cria um Característica do DataSource que lê dados de uma fonte como Amazon S3, JDBC ou o AWS Glue Data Catalog. Também oferece suporte a origens de dados de transmissão do Kafka e Kinesis.

- `connectionType`: o tipo da origem dos dados. Consulte the section called “Parâmetros de conexão”.
- `connectionOptions` – uma string de pares de nome/valor JSON que fornecem as informações adicionais para estabelecer uma conexão com a origem de dados. Para ter mais informações, consulte the section called “Parâmetros de conexão”.

Uma fonte de transmissão do Kinesis requer as seguintes opções de conexão: `streamARN`, `startingPosition`, `inferSchema` e `classification`.

Uma fonte de transmissão do Kafka requer as seguintes opções de conexão: `connectionName`, `topicName`, `startingOffsets`, `inferSchema` e `classification`.

- `transformationContext`: o contexto de transformação associado ao coletor a ser usado pelos marcadores de trabalho. Definido como vazio por padrão.
- `pushDownPredicate`: predicado em colunas de partição.

Retorna o `DataSource`.

Exemplo para fonte de transmissão do 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)
}
```

Exemplo para fonte de transmissão do 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
```

Cria um Característica do DataSource que lê dados de uma fonte como Amazon S3, JDBC ou AWS Glue Data Catalog e também define o formato dos dados armazenados na fonte.

- `connectionType`: o tipo da origem dos dados. Consulte the section called “Parâmetros de conexão”.
- `options`: uma string de pares de nome-valor JSON que fornecem as informações adicionais para estabelecer uma conexão com a origem dos dados. Consulte the section called “Parâmetros de conexão”.
- `transformationContext`: o contexto de transformação associado ao coletor a ser usado pelos marcadores de trabalho. Definido como vazio por padrão.
- `format`: o formato dos dados armazenados na fonte. Quando o `connectionType` é "s3", você também pode especificar `format`. Pode ser um de “avro”, “csv”, “grokLog”, “ion”, “json”, “xml”, “parquet” ou “orc”.
- `formatOptions`: uma string de pares de nome-valor JSON que fornecem opções adicionais para a análise dos dados na fonte. Consulte Opções de formato de dados.

Retorna o `DataSource`.

Exemplos

Crie um DynamicFrame a partir de uma fonte de dados que seja um arquivo de valores separados por vírgula (CSV) no 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()
```

Crie um DynamicFrame a partir de uma fonte de dados que seja um PostgreSQL usando uma conexão 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()
```

Crie um DynamicFrame a partir de uma fonte de dados que seja MySQL usando uma conexão 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()
```

definitivamente getSparkSession

```
def getSparkSession : SparkSession
```

Obtém o objeto `SparkSession` associado a este GlueContext. Use esse SparkSession objeto para registrar tabelas e UDFs para uso com `DataFrame` created from DynamicFrames.

Retorna SparkSession o.

def startTransaction

```
def startTransaction(readOnly: Boolean):String
```

Iniciar uma nova transação. Chama internamente a API startTransaction do Lake Formation.

- `readOnly`: (booleano) indica se esta transação deve ser somente de leitura ou de leitura e gravação. As gravações feitas usando um ID de transação somente de leitura serão rejeitadas. As transações somente de leitura não precisam ser confirmadas.

Retorna o ID da transação.

def commitTransaction

```
def commitTransaction(transactionId: String, waitForCommit: Boolean): Boolean
```

Tenta confirmar a transação especificada. `commitTransaction` pode retornar antes que a transação tenha terminado de confirmar. Chama internamente a API commitTransaction do Lake Formation.

- `transactionId`: (string) a transação a ser confirmada.
- `waitForCommit`: (booleano) Determina se `commitTransaction` retorna imediatamente. O valor padrão é true. Se for falso, `commitTransaction` sonda e aguarda até que a transação seja confirmada. A quantidade de tempo de espera é restrita a 1 minuto usando recuo exponencial com um máximo de 6 tentativas.

Retorna um booleano para indicar se a confirmação foi feita ou não.

def cancelTransaction

```
def cancelTransaction(transactionId: String): Unit
```

Tenta cancelar a transação especificada. Chama internamente a CancelTransactionAPI Lake Formation.

- `transactionId`: (string) a transação a ser cancelada.

Retorna uma exceção `TransactionCommittedException` se a transação tiver sido confirmada anteriormente.

def this

```
def this( sc : SparkContext,
          minPartitions : Int,
          targetPartitions : Int )
```

Cria um objeto `GlueContext` usando o `SparkContext` especificado, um mínimo de partições e partições de destino.

- `sc` — A ferramenta `SparkContext`.
- `minPartitions` – O número mínimo de partições.
- `targetPartitions` — O número destino de partições.

Retorna o `GlueContext`.

def this

```
def this( sc : SparkContext )
```

Cria um objeto `GlueContext` com o `SparkContext` fornecido. Define o mínimo de partições como 10 e as partições de destino como 20.

- `sc` — A ferramenta `SparkContext`.

Retorna o `GlueContext`.

def this

```
def this( sparkContext : JavaSparkContext )
```

Cria um objeto `GlueContext` com o `JavaSparkContext` fornecido. Define o mínimo de partições como 10 e as partições de destino como 20.

- `sparkContext` — A ferramenta `JavaSparkContext`.

Retorna o `GlueContext`.

MappingSpec

Pacote: com.amazonaws.services.glue

Classe do caso 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` – O `SchemaPath` do campo de origem.
- `sourceType` – O `DataType` do campo de origem.
- `targetPath` – O `SchemaPath` do campo de destino.
- `targetType` – O `DataType` do campo de destino.

Um `MappingSpec` especifica um mapeamento de um caminho de origem e um tipo de dados de origem para um caminho de destino e um tipo de dados de destino. O valor no caminho de origem no quadro de origem aparece no quadro de destino no caminho de destino. O tipo de dados de origem é convertido no tipo de dados de destino.

Ele se estende a partir de `Product4` para que você possa lidar com qualquer `Product4` em sua interface do `applyMapping`.

Objeto MappingSpec

```
object MappingSpec
```

O objeto `MappingSpec` tem os seguintes membros:

Val orderingByTarget

```
val orderingByTarget: Ordering[MappingSpec]
```

Def apply

```
def apply( sourcePath : String,
           sourceType : DataType,
           targetPath : String,
           targetType : DataType
         ) : MappingSpec
```

Cria um `MappingSpec`.

- `sourcePath` – Uma representação de string do caminho de origem.
- `sourceType` — O `DataType` de origem.
- `targetPath` – Uma representação de string do caminho de destino.
- `targetType` - O `DataType` de destino.

Retorna um `MappingSpec`.

Def apply

```
def apply( sourcePath : String,
           sourceTypeString : String,
           targetPath : String,
           targetTypeString : String
         ) : MappingSpec
```

Cria um `MappingSpec`.

- `sourcePath` – Uma representação de string do caminho de origem.
- `sourceType` – Uma representação de string do tipo de dados de origem.
- `targetPath` – Uma representação de string do caminho de destino.
- `targetType` – Uma representação de string do tipo de dados de destino.

Retorna um MappingSpec.

Def apply

```
def apply( product : Product4[String, String, String, String] ) : MappingSpec
```

Cria um `MappingSpec`.

- `product` — O `Product4` do caminho de origem, do tipo de dados de origem, do caminho de destino e do tipo de dados de destino.

Retorna um `MappingSpec`.

APIs ResolveSpec no Scala do AWS Glue

Tópicos

- Objeto ResolveSpec
- Classe do caso ResolveSpec

Pacote: com.amazonaws.services.glue

Objeto ResolveSpec

ResolveSpec

```
object ResolveSpec
```

Def

```
def apply( path : String,
           action : String
         ) : ResolveSpec
```

Cria um `ResolveSpec`.

- `path` – Uma representação da string do campo de escolha que precisa ser resolvido.
- `action`: uma ação de resolução. A ação pode ser uma das seguintes: `Project`, `KeepAsStruct` ou `Cast`.

Retorna o `ResolveSpec`.

Def

```
def apply( product : Product2[String, String] ) : ResolveSpec
```

Cria um `ResolveSpec`.

- `product` — `Product2` de: caminho de origem, ação de resolução.

Retorna o `ResolveSpec`.

Classe do caso ResolveSpec

```
case class ResolveSpec extends Product2[String, String]  (
           path : SchemaPath,
           action : String )
```

Cria um `ResolveSpec`.

- `path` – O `SchemaPath` do campo de escolha que precisa ser resolvido.
- `action`: uma ação de resolução. A ação pode ser uma das seguintes: `Project`, `KeepAsStruct` ou `Cast`.

Métodos def ResolveSpec

```
def _1 : String
```

```
def _2 : String
```

APIs ArrayNode em Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso 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 )
```

APIs BinaryNode em Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso BinaryNode

BinaryNode

```
case class BinaryNode extends ScalarNode(value, TypeCode.BINARY)  (
           value : Array[Byte] )
```

Campos val BinaryNode

- `ordering`

Métodos def BinaryNode

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

APIs BooleanNode em Scala no AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso BooleanNode

BooleanNode

```
case class BooleanNode extends ScalarNode(value, TypeCode.BOOLEAN)  (
           value : Boolean )
```

Campos val BooleanNode

- `ordering`

Métodos def BooleanNode

```
def equals( other : Any )
```

APIs de ByteNode de Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso ByteNode

ByteNode

```
case class ByteNode extends ScalarNode(value, TypeCode.BYTE)  (
```

```
            value : Byte )
```

Campos val ByteNode

- `ordering`

Métodos def ByteNode

```
def equals( other : Any )
```

APIs de DateNode de Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso DateNode

DateNode

```
case class DateNode extends ScalarNode(value, TypeCode.DATE)  (
            value : Date )
```

Campos val DateNode

- `ordering`

Métodos def DateNode

```
def equals( other : Any )
```

```
def this( value : Int )
```

APIs DecimalNode em Scala no AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso DecimalNode

DecimalNode

```
case class DecimalNode extends ScalarNode(value, TypeCode.DECIMAL)  (
```

```
            value : BigDecimal )
```

Campos val DecimalNode

- `ordering`

Métodos def DecimalNode

```
def equals( other : Any )
```

```
def this( value : Decimal )
```

APIs DoubleNode em Scala no AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso DoubleNode

DoubleNode

```
case class DoubleNode extends ScalarNode(value, TypeCode.DOUBLE)  (
            value : Double )
```

Campos val DoubleNode

- `ordering`

Métodos def DoubleNode

```
def equals( other : Any )
```

APIs DynamicNode em Scala no AWS Glue

Tópicos

- Classe DynamicNode
- Objeto DynamicNode

Pacote: com.amazonaws.services.glue.types

Classe DynamicNode

DynamicNode

```
class DynamicNode extends Serializable with Cloneable
```

Métodos def DynamicNode

```
def getValue : Any
```

Obter um valor simples e vinculado ao registro atual:

```
def nodeType : TypeCode
```

```
def toJson : String
```

Método para depuração:

```
def toRow( schema : Schema,
           options : Map[String, ResolveOption]
         ) : Row
```

```
def typeName : String
```

Objeto DynamicNode

DynamicNode

```
object DynamicNode
```

Métodos def DynamicNode

```
def quote( field : String,
           useQuotes : Boolean
         ) : String
```

```
def quote( node : DynamicNode,
           useQuotes : Boolean
         ) : String
```

Classe EvaluateDataQuality

O AWS Glue Data Quality está em uma versão pré-lançamento do AWS Glue e está sujeito a alterações.

Pacote: com.amazonaws.services.glue.dq

```
object EvaluateDataQuality
```

Def apply

```
def apply(frame: DynamicFrame,
            ruleset: String,
            publishingOptions: JsonOptions = JsonOptions.empty): DynamicFrame
```

Avalia um conjunto de regras de qualidade de dados em relação a um `DynamicFrame` e retorna um novo `DynamicFrame` com os resultados da avaliação. Para saber mais sobre o AWS Glue Data Quality, consulte AWS Glue Qualidade de dados.

- `frame`: o `DynamicFrame` cuja qualidade de dados você deseja avaliar.
- `ruleset`: um conjunto de regras em Data Quality Definition Language (DQDL) no formato de string. Para saber mais sobre DQDL, consulte o guia de Referência de Data Quality Definition Language (DQDL).
- `publishingOptions`: um dicionário que especifica as seguintes opções para publicar resultados e métricas de avaliação:
  - `dataQualityEvaluationContext`: uma string que especifica o namespace com o qual o AWS Glue deve publicar as métricas e os resultados de qualidade dos dados do Amazon CloudWatch. As métricas agregadas aparecem no CloudWatch, enquanto os resultados completos aparecem na interface do AWS Glue Studio.
    - Obrigatório: Não
    - Valor padrão: `default_context`
  - `enableDataQualityCloudWatchMetrics`: especifica se os resultados da avaliação de qualidade dos dados devem ser publicados no CloudWatch. Você especifica um namespace para as métricas usando a opção `dataQualityEvaluationContext`.
    - Obrigatório: Não

  - Valor padrão: False
- `enableDataQualityResultsPublishing`: especifica se os resultados de qualidade dos dados devem estar visíveis na guia Data Quality (Qualidade de dados) na interface do AWS Glue Studio.
  - Obrigatório: Não
  - Valor padrão: true
- `resultsS3Prefix`: especifica o local no Amazon S3 em que o AWS Glue pode gravar os resultados da avaliação de qualidade dos dados.
  - Obrigatório: Não
  - Valor padrão: "" (string vazia)

Exemplo

O código de exemplo a seguir demonstra como avaliar a qualidade dos dados de um `DynamicFrame` antes de realizar uma transformação `SelectFields`. O script verifica se todas as regras de qualidade de dados são aprovadas antes de tentar a transformação.

```
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()
  }
}
```

APIs FloatNode em Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso FloatNode

FloatNode

```
case class FloatNode extends ScalarNode(value, TypeCode.FLOAT)  (
           value : Float )
```

Campos val FloatNode

- `ordering`

Métodos def FloatNode

```
def equals( other : Any )
```

Classe FillMissingValues

Pacote: 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
```

Preenche os valores ausentes de um quadro dinâmico em uma coluna especificada e retorna um novo quadro com estimativas em uma nova coluna. Para linhas sem valores ausentes, o valor da coluna especificada é duplicado para a nova coluna.

- `frame`: o DynamicFrame no qual preencher valores ausentes. Obrigatório.
- `missingValuesColumn`: a coluna que contém valores ausentes (valores `null` e strings vazias). Obrigatório.
- `outputColumn`: o nome da nova coluna que conterá valores estimados para todas as linhas cujo valor estava ausente. Opcional. O padrão é o valor da `missingValuesColumn` seguida do sufixo `"_filled"`.
- `transformationContext`: uma string única que é usada para identificar informações de estado (opcional).
- `callSite`: usado para fornecer informações de contexto para o relatório de erros (opcional).
- `stageThreshold`: o número máximo de erros que podem ocorrer na transformação antes que ela falhe (opcional, o padrão é zero).
- `totalThreshold`: o número máximo de erros que podem ocorrer antes que o processamento falhe (opcional, o padrão é zero).

Retorna um novo quadro dinâmico com uma coluna adicional que contém estimativas para linhas com valores ausentes e o valor presente para outras linhas.

Classe FindMatches

Pacote: 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
```

Localize correspondências em um quadro de entrada e retorne um novo quadro com uma nova coluna contendo um ID exclusivo por grupo de correspondência.

- `frame`: o DynamicFrame no qual encontrar correspondências. Obrigatório.
- `transformId`: o ID exclusivo associado à transformação FindMatches a ser aplicado em registros no quadro de entrada. Obrigatório.
- `transformationContext`: o identificador para esse `DynamicFrame`. O `transformationContext` é usado como uma chave para o estado do marcador de trabalho que é mantido nas execuções. Opcional.
- `callSite`: usado para fornecer informações de contexto para o relatório de erros. Esses valores são definidos automaticamente quando chamados do Python. Opcional.
- `stageThreshold`: o número máximo de registros de erros permitidos a partir da computação desse `DynamicFrame` antes de lançar uma exceção, excluindo os registros presentes no `DynamicFrame` anterior. Opcional. O padrão é zero.
- `totalThreshold`: o número máximo de registros de erros totais antes que uma exceção seja lançada, incluindo os de quadros anteriores. Opcional. O padrão é zero.
- `enforcedMatches`: o quadro para correspondências impostas. Opcional. O padrão é `null`.
- `computeMatchConfidenceScores`: um valor booleano que indica se é preciso calcular uma pontuação de confiança para cada grupo de registros correspondentes. Opcional. O padrão é falso.

Retorna um novo quadro dinâmico com um identificador exclusivo atribuído a cada grupo de registros correspondentes.

Classe FindIncrementalMatches

Pacote: 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
```

Encontre correspondências entre os quadros existentes e incrementais e retorne um novo quadro com uma coluna contendo um ID exclusivo por grupo de correspondência.

- `existingframe`: um quadro existente ao qual foi atribuído um ID correspondente para cada grupo. Obrigatório.
- `incrementalframe`: um quadro incremental usado para encontrar correspondências no quadro existente. Obrigatório.
- `transformId`: o ID exclusivo associado à transformação FindIncrementalMatches a ser aplicado em registros no quadro de entrada. Obrigatório.
- `transformationContext`: o identificador para esse `DynamicFrame`. O `transformationContext` é usado como uma chave para o estado do marcador de trabalho que é mantido nas execuções. Opcional.
- `callSite`: usado para fornecer informações de contexto para o relatório de erros. Esses valores são definidos automaticamente quando chamados do Python. Opcional.
- `stageThreshold`: o número máximo de registros de erros permitidos a partir da computação desse `DynamicFrame` antes de lançar uma exceção, excluindo os registros presentes no `DynamicFrame` anterior. Opcional. O padrão é zero.

- `totalThreshold`: o número máximo de registros de erros totais antes que uma exceção seja lançada, incluindo os de quadros anteriores. Opcional. O padrão é zero.
- `enforcedMatches`: o quadro para correspondências impostas. Opcional. O padrão é `null`.
- `computeMatchConfidenceScores`: um valor booleano que indica se é preciso calcular uma pontuação de confiança para cada grupo de registros correspondentes. Opcional. O padrão é falso.

Retorna um novo quadro dinâmico com um identificador exclusivo atribuído a cada grupo de registros correspondentes.

APIs de IntegerNode de Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso IntegerNode

IntegerNode

```
case class IntegerNode extends ScalarNode(value, TypeCode.INT)  (
           value : Int )
```

Campos val IntegerNode

- `ordering`

Métodos def IntegerNode

```
def equals( other : Any )
```

APIs LongNode em Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso LongNode

LongNode

```
case class LongNode extends ScalarNode(value, TypeCode.LONG)  (
           value : Long )
```

Campos val LongNode

- `ordering`

Métodos def LongNode

```
def equals( other : Any )
```

APIs MapLikeNode no Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe MapLikeNode

MapLikeNode

```
class MapLikeNode extends DynamicNode  (
           value : mutable.Map[String, DynamicNode] )
```

Métodos def 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
```

Exemplo: Dado esse JSON:

```
{"foo": "bar"}
```

Se `useQuotes == true`, `toJson` gera `{"foo": "bar"}`. Se `useQuotes == false`, `toJson` gera `{foo: bar}` @return.

APIs MapNode no Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso MapNode

MapNode

```
case class MapNode extends MapLikeNode(value)  (
           value : mutable.Map[String, DynamicNode] )
```

Métodos def MapNode

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

```
def nodeType
```

```
def this
```

APIs NullNode no Scala do AWS Glue

Tópicos

Pacote: com.amazonaws.services.glue.types

Classe NullNode

NullNode

```
class NullNode
```

Objeto do caso NullNode

NullNode

```
case object NullNode extends NullNode
```

APIs ObjectNode no Scala do AWS Glue

Tópicos

Pacote: com.amazonaws.services.glue.types

Objeto ObjectNode

ObjectNode

```
object ObjectNode
```

Métodos def ObjectNode

```
def apply( frameKeys : Set[String],
           v1 : mutable.Map[String, DynamicNode],
           v2 : mutable.Map[String, DynamicNode],
```

```
            resolveWith : String
          ) : ObjectNode
```

Classe do caso ObjectNode

ObjectNode

```
case class ObjectNode extends MapLikeNode(value)  (
            val value : mutable.Map[String, DynamicNode] )
```

Métodos def ObjectNode

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

```
def nodeType
```

```
def this
```

APIs ScalarNode no Scala do AWS Glue

Tópicos

- Classe ScalarNode
- Objeto ScalarNode

Pacote: com.amazonaws.services.glue.types

Classe ScalarNode

ScalarNode

```
class ScalarNode extends DynamicNode  (
            value : Any,
            scalarType : TypeCode )
```

Métodos def ScalarNode

```
def compare( other : Any,
             operator : String
           ) : Boolean
```

```
def getValue
```

```
def hashCode : Int
```

```
def nodeType
```

```
def toJson
```

Objeto ScalarNode

ScalarNode

```
object ScalarNode
```

Métodos def 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
```

APIs ShortNode no Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso ShortNode

ShortNode

```
case class ShortNode extends ScalarNode(value, TypeCode.SHORT)  (
           value : Short )
```

Campos val ShortNode

- `ordering`

Métodos def ShortNode

```
def equals( other : Any )
```

APIs StringNode no Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso StringNode

StringNode

```
case class StringNode extends ScalarNode(value, TypeCode.STRING)  (
           value : String )
```

Campos val StringNode

- `ordering`

Métodos def StringNode

```
def equals( other : Any )
```

```
def this( value : UTF8String )
```

APIs TimestampNode no Scala do AWS Glue

Pacote: com.amazonaws.services.glue.types

Classe do caso TimestampNode

TimestampNode

```
case class TimestampNode extends ScalarNode(value, TypeCode.TIMESTAMP)  (
           value : Timestamp )
```

Campos val TimestampNode

- `ordering`

Métodos def TimestampNode

```
def equals( other : Any )
```

```
def this( value : Long )
```

APIs GlueArgParser em Scala no AWS Glue

Pacote: com.amazonaws.services.glue.util

Objeto GlueArgParser

GlueArgParser

```
object GlueArgParser
```

Isso é estritamente consistente com a versão do Python de `utils.getResolvedOptions` no pacote `AWSGlueDataplanePython`.

Métodos def GlueArgParser

```
def getResolvedOptions( args : Array[String],
                        options : Array[String]
                      ) : Map[String, String]
```

```
def initParser( userOptionsSet : mutable.Set[String] ) : ArgumentParser
```

Example Recuperar argumentos passados para um trabalho

Para recuperar os argumentos do trabalho, você pode usar o método `getResolvedOptions`. Considere o exemplo a seguir, que recupera um argumento de trabalho 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)
```

APIs de trabalho em Scala do AWS Glue

Pacote: com.amazonaws.services.glue.util

Objeto Job

Trabalho

```
object Job
```

Métodos def Job

```
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
```

Recursos e otimizações para programação de scripts de ETL do AWS Glue para Spark

As seções a seguir descrevem técnicas e valores que se aplicam geralmente à programação de ETL (extração, transformação e carregamento) do AWS Glue para Spark em qualquer linguagem.

Tópicos

Tipos e opções de conexão para ETL no AWS Glue para Spark

No AWS Glue Glue, vários métodos e transformações do PySpark e do Scala especificam o tipo de conexão usando um parâmetro `connectionType`. Eles especificam opções de conexão usando um parâmetro `connectionOptions` ou `options`.

O parâmetro `connectionType` pode usar os valores mostrados na tabela a seguir. Os valores do parâmetro `connectionOptions` (ou `options`) associados para cada tipo estão documentados nas seções a seguir. Salvo indicação em contrário, os parâmetros se aplicam quando a conexão é usada como fonte ou coletor.

Para obter um código de exemplo que demonstra a configuração e o uso de opções de conexão, consulte a página inicial de cada tipo de conexão.

`connectionType`	Conecta-se a
dynamodb	Banco de dados do Amazon DynamoDB
kinesis	Amazon Kinesis Data Streams

`connectionType`	Conecta-se a
s3	Amazon S3
documentdb	banco de dados do Amazon DocumentDB (compatível com MongoDB)
opensearch	Amazon OpenSearch Service.
redshift	Banco de dados do Amazon Redshift
kafka	Kafka ou Amazon Managed Streaming for Apache Kafka
azurecosmos	Azure Cosmos para NoSQL.
azuresql	Azure SQL.
bigquery	Google BigQuery.
mongodb	Banco de dados MongoDB, incluindo MongoDB Atlas.
sqlserver	Banco de dados do Microsoft SQL Server (consulte Conexões JDBC)
mysql	Banco de dados MySQL (consulte Conexões JDBC)
oracle	Banco de dados Oracle (consulte Conexões JDBC)
postgresql	Banco de dados PostgreSQL (consulte Conexões JDBC)
saphana	SAP HANA.
snowflake	Data lake do Snowflake
teradata	Teradata Vantage.
vertica	Vertica.
custom.*	Armazenamentos de dados do Spark, Athena ou JDBC (consulte Valores de connectionType personalizados e AWS Marketplace
marketplace.*	Armazenamentos de dados do Spark, Athena ou JDBC (consulte Valores de connectionType personalizados e AWS Marketplace)

Conexões do DynamoDB

É possível usar o AWS Glue para Spark para ler e gravar em tabelas no DynamoDB no AWS Glue. Conecte-se ao DynamoDB usando as permissões do IAM anexadas ao seu trabalho do AWS Glue. O AWS Glueoferece suporte à gravação de dados em uma tabela do DynamoDB em uma conta da AWS diferente. Para ter mais informações, consulte the section called “Acesso a tabelas do DynamoDB entre contas e entre regiões”.

Além do conector de ETL para AWS Glue DynamoDB, é possível ler do DynamoDB usando o conector de exportação para DynamoDB. Esse conector invoca uma solicitação `ExportTableToPointInTime` do DynamoDB e a armazena no formato DynamoDB JSON em um local do Amazon S3 fornecido por você. Em seguida, o AWS Glue cria um objeto DynamicFrame ao ler os dados do local de exportação do Amazon S3.

O gravador do DynamoDB é compatível com o AWS Glue versão 1.0 ou posterior. O conector de exportação do AWS Glue DynamoDB é compatível com o AWS Glue versão 2.0 ou posterior.

Para obter mais informações sobre o DynamoDB, consulte a documentação do Amazon DynamoDB.

> **Note**
>
> O leitor de ETL do DynamoDB não é compatível com filtros ou predicados de aplicação.

Configurar conexões do DynamoDB

Para se conectar ao DynamoDB via AWS Glue, conceda ao perfil do IAM associado ao seu trabalho do AWS Glue permissão para interagir com o DynamoDB. Para obter mais informações sobre as permissões necessárias para ler ou gravar no DynamoDB, consulte Ações, recursos e chaves de condição para DynamoDB na documentação do IAM.

Nas seguintes situações, configurações adicionais podem ser necessárias:

- Ao usar o conector de exportação do DynamoDB, você precisará configurar o IAM para que seu trabalho possa solicitar exportações de tabelas do DynamoDB. Além disso, você precisará identificar um bucket do Amazon S3 para a exportação e fornecer as permissões apropriadas no IAM para que o DynamoDB grave nele e para que seu trabalho do AWS Glue possa ler dele. Para obter mais informações, consulte Solicitar uma exportação de tabela no DynamoDB.
- Se sua tarefa do AWS Glue tiver requisitos específicos de conectividade da Amazon VPC, use o tipo de conexão `NETWORK` do AWS Glue para fornecer opções de rede. Como o acesso

ao DynamoDB é autorizado pelo IAM, não há necessidade de um tipo de conexão AWS Glue DynamoDB.

Ler e gravar no DynamoDB

Os exemplos de código a seguir mostram como fazer a leitura (via conector ETL) e gravação de tabelas do DynamoDB. Eles demonstram a leitura de uma tabela e a gravação em uma outra tabela.

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()
  }

}
```

Usar o conector de exportação do DynamoDB

O conector de exportação apresenta um melhor desempenho do que o conector de ETL quando o tamanho da tabela do DynamoDB é superior a 80 GB. Além disso, como a solicitação de exportação é conduzida fora dos processos do Spark em um trabalho do AWS Glue, você pode habilitar a autoescalabilidade de trabalhos do AWS Glue para poupar o uso de DPU durante a solicitação de exportação. Com o conector de exportação, você também não precisa configurar o número de divisões para o paralelismo do executor do Spark ou o percentual de leitura de throughput do DynamoDB.

> **Note**
>
> O DynamoDB tem requisitos específicos para invocar as solicitações `ExportTableToPointInTime`. Para mais informações, consulte Solicitação de uma exportação de tabela no DynamoDB. Por exemplo, para usar esse conector, é necessário que a Point-in-Time-Recovery (PITR – Recuperação em um ponto anterior no tempo) esteja habilitada na tabela. O conector do DynamoDB também é compatível com criptografia do AWS KMS para exportações do DynamoDB para o Amazon S3. O fornecimento de sua configuração de segurança na configuração do trabalho do AWS Glue habilita a criptografia do AWS KMS para uma exportação do DynamoDB. A chave do KMS precisa estar na mesma região do bucket do Amazon S3.
> Observe que há cobranças adicionais para exportação do DynamoDB e custos de armazenamento do Amazon S3. Os dados exportados no Amazon S3 persistem após a conclusão de uma execução de trabalho, de modo que você possa reutilizá-los sem exportações adicionais do DynamoDB. Um requisito para o uso desse conector é que a recuperação a um ponto anterior no tempo (PITR) esteja habilitada para a tabela.
> O conector de ETL e o conector de exportação do DynamoDB não são compatíveis com a aplicação de filtros ou predicados de aplicação na fonte do DynamoDB.

Os exemplos de código a seguir mostram como fazer a leitura (usando o conector de exportação) e imprimir o número de partições.

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()
  }

}
```

Estes exemplos mostram como fazer a leitura (usando o conector de exportação) e imprimir o número de partições de uma tabela do AWS Glue Data Catalog que tenha uma classificação 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())
)
```

Simplificar o uso do JSON de exportação do DynamoDB

As exportações do DynamoDB feitas com o conector de exportação para DynamoDB do AWS Glue resultam em arquivos JSON de estruturas aninhadas específicas. Para mais informações, consulte Objetos de dados. O AWS Glue fornece uma transformação DynamicFrame, que pode desaninhar essas estruturas em uma forma mais fácil de usar para aplicações downstream.

A transformação pode ser invocada de duas formas. É possível definir a opção de conexão `"dynamodb.simplifyDDBJson"` com o valor `"true"` ao chamar um método para ler do

DynamoDB. Você também pode chamar a transformação como um método disponível de forma independente na biblioteca do AWS Glue.

Considere o seguinte esquema gerado por uma exportação do 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
```

A transformação `simplifyDDBJson` simplificará isso para:

```
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` só está disponível no AWS Glue 3.0 e versões posteriores. A transformação `unnestDDBJson` também está disponível para simplificar o JSON de exportação do DynamoDB. Incentivamos os usuários a fazer a transição de `unnestDDBJson` para `simplifyDDBJson`.

Configurar o paralelismo nas operações do DynamoDB

Para aumentar a performance, é possível ajustar alguns parâmetros disponíveis para o conector do DynamoDB. Seu objetivo ao ajustar os parâmetros de paralelismo é maximizar o uso dos operadores do AWS Glue provisionados. Assim, se você precisar de mais performance, recomendamos aumentar a escala do seu trabalho horizontalmente aumentando o número de DPUs.

Você pode alterar o paralelismo em uma operação de leitura do DynamoDB usando o parâmetro `dynamodb.splits` ao utilizar o conector ETL. Ao ler com o conector de exportação, não é necessário configurar o número de divisões para o paralelismo do executor do Spark. Você pode alterar o paralelismo em uma operação de gravação do DynamoDB com `dynamodb.output.numParallelTasks`.

Ler com o conector ETL do DynamoDB

Recomendamos calcular `dynamodb.splits` com base no número máximo de trabalhadores definido em sua configuração de trabalho e no cálculo `numSlots` a seguir. Em caso de escalonamento automático, o número real de operadores disponíveis pode cair abaixo desse limite. Para obter mais informações sobre como definir o número máximo de trabalhadores, consulte Número de trabalhadores (`NumberOfWorkers`) em the section called “Configurando as propriedades da tarefa do Spark”.

- `numExecutors = NumberOfWorkers - 1`

  Para contextualizar, um executor é reservado para o driver do Spark, outros executores são usados para processar dados.

- `numSlotsPerExecutor =`

  AWS Glue 3.0 and later versions

  - 4 se `WorkerType` for `G.1X`
  - 8 se `WorkerType` for `G.2X`
  - 16 se `WorkerType` for `G.4X`
  - 32 se `WorkerType` for `G.8X`

  AWS Glue 2.0 and legacy versions

  - 8 se `WorkerType` for `G.1X`
  - 16 se `WorkerType` for `G.2X`
- `numSlots = numSlotsPerExecutor * numExecutors`

Recomendamos definir `dynamodb.splits` como o número de slots disponíveis, `numSlots`.

Gravar no DynamoDB

O parâmetro `dynamodb.output.numParallelTasks` é usado para determinar a WCU por tarefa do Spark, usando o seguinte cálculo:

```
permittedWcuPerTask = ( TableWCU * dynamodb.throughput.write.percent ) / 
dynamodb.output.numParallelTasks
```

O gravador do DynamoDB funcionará melhor se a configuração representar com precisão o número de tarefas do Spark gravadas no DynamoDB. Em alguns casos, talvez seja necessário substituir o cálculo padrão para aumentar o desempenho de gravação. Se você não especificar esse parâmetro, as WCU permitidas por tarefa do Spark serão calculadas automaticamente pela seguinte fórmula:

-
  - `numPartitions = dynamicframe.getNumPartitions()`
  - `numSlots` (conforme definido anteriormente nesta seção)
  - `numParallelTasks = min(numPartitions, numSlots)`
- Exemplo 1. DPU=10, WorkerType=Standard. DynamicFrame de entrada tem 100 partições RDD.
  - `numPartitions = 100`
  - `numExecutors = (10 - 1) * 2 - 1 = 17`
  - `numSlots = 4 * 17 = 68`
  - `numParallelTasks = min(100, 68) = 68`
- Exemplo 2. DPU=10, WorkerType=Standard. DynamicFrame de entrada tem 20 partições RDD.

- `numPartitions = 20`
- `numExecutors = (10 - 1) * 2 - 1 = 17`
- `numSlots = 4 * 17 = 68`
- `numParallelTasks = min(20, 68) = 20`

> **Note**
>
> Trabalhos em versões antigas do AWS Glue e aqueles que usam operadores Standard exigem métodos diferentes para calcular o número de slots. Se você precisar ajustar a performance desses trabalhos, recomendamos fazer a transição para as versões compatíveis do AWS Glue.

Referência de opções de conexão do DynamoDB

Designa uma conexão com o Amazon DynamoDB.

As opções de conexão diferem para uma conexão da fonte e uma conexão do coletor.

“connectionType”: “dynamodb” com o conector de ETL como fonte

Ao usar o conector de ETL para DynamoDB do AWS Glue, use as seguintes opções de conexão com `"connectionType": "dynamodb"` como fonte:

- `"dynamodb.input.tableName"`: (obrigatório) a tabela do DynamoDB da qual fazer a leitura.
- `"dynamodb.throughput.read.percent"`: (opcional) a porcentagem de unidades de capacidade de leitura (RCU) para usar. O padrão é definido como "0,5". Os valores aceitáveis são de "0,1" a "1,5", inclusive.
  - `0.5` representa a taxa de leitura padrão, o que significa que o AWS Glue tentará consumir metade da capacidade de leitura da tabela. Se você aumentar o valor acima de `0.5`, o AWS Glue aumentará a taxa de solicitação. Diminuir o valor abaixo de `0.5` reduz a taxa de solicitação de leitura. (A taxa de leitura real poderá variar, dependendo de fatores como se há uma distribuição de chaves uniformes na tabela do DynamoDB.)
  - Quando a tabela do DynamoDB está no modo sob demanda, o AWS Glue lida com a capacidade de leitura da tabela como 40.000. Para exportar uma tabela grande, recomendamos alternar sua tabela do DynamoDB para o modo sob demanda.

- `"dynamodb.splits"`: (opcional) define em quantas divisões essa tabela do DynamoDB deve ser particionada ao fazer a leitura. O padrão é definido como "1". Os valores aceitáveis são de "1" a "1,000,000", inclusive.

  1 indica que não há paralelismo. É altamente recomendável que você especifique um valor maior para uma performance melhor usando a fórmula abaixo. Para obter mais informações sobre como definir adequadamente um valor, consulte the section called “Paralelismo dinâmico”.
- `"dynamodb.sts.roleArn"`: (opcional) o ARN da função do IAM a ser assumida para acesso entre contas. Esse parâmetro está disponível no AWS Glue 1.0 ou posterior.
- `"dynamodb.sts.roleSessionName"`: (opcional) nome da sessão STS. O padrão é definido como “glue-dynamodb-read-sts-session”. Esse parâmetro está disponível no AWS Glue 1.0 ou posterior.

“connectionType”: “dynamodb” com o conector de exportação para DynamoDB do AWS Glue como fonte

Use as seguintes opções de conexão com "connectionType": "dynamodb" como fonte ao usar o conector de exportação para DynamoDB do AWS Glue, que está disponível apenas para o AWS Glue versão 2.0 em diante:

- `"dynamodb.export"`: (obrigatório) um valor de string:
  - Se definido como `ddb`, habilita o conector de exportação para DynamoDB do AWS Glue. Um novo `ExportTableToPointInTimeRequest` será invocado durante o trabalho do AWS Glue. Uma nova exportação será gerada com o local repassado de `dynamodb.s3.bucket` e `dynamodb.s3.prefix`.
  - Se definido como `s3`, habilita o conector de exportação para DynamoDB do AWS Glue, mas ignora a criação de uma nova exportação do DynamoDB. Em vez disso, usa o `dynamodb.s3.bucket` e `dynamodb.s3.prefix` como o local do Amazon S3 de uma exportação anterior dessa tabela.
- `"dynamodb.tableArn"`: (obrigatório) a tabela do DynamoDB da qual fazer a leitura.
- `"dynamodb.unnestDDBJson"`: (Opcional) Padrão: false. Valores válidos: booleano. Se definido como “true” (verdadeiro), executa uma transformação de desaninhamento da estrutura JSON do DynamoDB que está presente nas exportações. É um erro definir `"dynamodb.unnestDDBJson"` e `"dynamodb.simplifyDDBJson"` como verdadeiro ao mesmo tempo. No AWS Glue 3.0 e versões posteriores, recomendamos usar `"dynamodb.simplifyDDBJson"` para melhorar

o comportamento ao simplificar os tipos de mapas do DynamoDB. Para ter mais informações, consulte the section called “Simplificar o uso do JSON de exportação do DynamoDB”.

- `"dynamodb.simplifyDDBJson"`: (Opcional) Padrão: false. Valores válidos: booleano. Se definido como “true” (verdadeiro), executa uma transformação para simplificar o esquema da estrutura JSON do DynamoDB que está presente nas exportações. Isso tem a mesma finalidade que a opção `"dynamodb.unnestDDBJson"`, mas fornece melhor suporte a tipos de mapas do DynamoDB ou até mesmo tipos de mapas aninhados em sua tabela do DynamoDB. Esse atributo só está disponível no AWS Glue 3.0 e versões posteriores. É um erro definir `"dynamodb.unnestDDBJson"` e `"dynamodb.simplifyDDBJson"` como verdadeiro ao mesmo tempo. Para ter mais informações, consulte the section called “Simplificar o uso do JSON de exportação do DynamoDB”.
- `"dynamodb.s3.bucket"`: (opcional) indica o local do bucket do Amazon S3 no qual o processo `ExportTableToPointInTime` do DynamoDB deve ser executado. O formato de arquivo para a exportação é DynamoDB JSON.
  - `"dynamodb.s3.prefix"`: (Opcional) indica o local do prefixo do Amazon S3 dentro do bucket do Amazon S3 no qual as cargas `ExportTableToPointInTime` do DynamoDB devem ser armazenadas. Se não houver a especificação de `dynamodb.s3.prefix` e `dynamodb.s3.bucket`, esses valores serão definidos por padrão para o local do diretório temporário especificado na configuração de trabalho do AWS Glue. Para mais informações, consulte Parâmetros especiais usados pelo AWS Glue.
  - `"dynamodb.s3.bucketOwner"`: indica o proprietário do bucket necessário para acesso entre contas do Amazon S3.
- `"dynamodb.sts.roleArn"`: (opcional) o ARN do perfil do IAM a ser assumido para acesso entre contas e/ou acesso entre regiões para a tabela do DynamoDB. Observação: o mesmo ARN de função do IAM será usado para acessar o local do Amazon S3 especificado para a solicitação `ExportTableToPointInTime`.
- `"dynamodb.sts.roleSessionName"`: (opcional) nome da sessão STS. O padrão é definido como “glue-dynamodb-read-sts-session”.
- `"dynamodb.exportTime"` (Opcional) Valores válidos: strings representando instantes ISO-8601. Um momento no qual a exportação deve ser feita.
- `"dynamodb.sts.region"`: (obrigatório se estiver fazendo uma chamada entre regiões usando um endpoint regional) a região que hospeda a tabela do DynamoDB que você deseja ler.

“connectionType”: “dynamodb” com o conector de ETL como coletor

Use as seguintes opções de conexão com `"connectionType": "dynamodb"` como coletor:

- `"dynamodb.output.tableName"`: (obrigatório) a tabela do DynamoDB na qual fazer a gravação.
- `"dynamodb.throughput.write.percent"`: (opcional) a porcentagem de unidades de capacidade de gravação (WCU) para usar. O padrão é definido como "0,5". Os valores aceitáveis são de "0,1" a "1,5", inclusive.
  - `0.5` representa a taxa de gravação padrão, o que significa que o AWS Glue tentará consumir metade da capacidade de gravação da tabela. Se você aumentar o valor acima de 0,5, o AWS Glue aumentará a taxa de solicitação. Diminuir o valor abaixo de 0,5 reduz a taxa de solicitação de gravação. (A taxa de gravação real pode variar, dependendo de fatores como a existência de uma distribuição de chaves uniforme na tabela do DynamoDB.)
  - Quando a tabela do DynamoDB está no modo sob demanda, o AWS Glue lida com a capacidade de gravação da tabela como `40000`. Para importar uma tabela grande, recomendamos mudar sua tabela do DynamoDB para o modo sob demanda.
- `"dynamodb.output.numParallelTasks"`: (opcional) define quantas tarefas simultâneas gravam no DynamoDB ao mesmo tempo. Usado para calcular as WCU permissivas por tarefa do Spark. Na maioria dos casos, o AWS Glue calculará um valor padrão razoável para esse valor. Para ter mais informações, consulte the section called “Paralelismo dinâmico”.
- `"dynamodb.output.retry"`: (opcional) define quantas novas tentativas realizamos quando há uma `ProvisionedThroughputExceededException` do DynamoDB. O padrão é definido como “10”.
- `"dynamodb.sts.roleArn"`: (opcional) o ARN da função do IAM a ser assumida para acesso entre contas.
- `"dynamodb.sts.roleSessionName"`: (opcional) nome da sessão STS. O padrão é definido como “glue-dynamodb-write-sts-session”.

Acesso a tabelas do DynamoDB entre contas e entre regiões

Os trabalhos de ETL do AWS Glue oferecem suporte a acesso a tabelas do DynamoDB tanto entre regiões quanto entre contas. Os trabalhos de ETL do AWS Glue suportam tanto a leitura de dados de uma tabela do DynamoDB de outra conta da AWS quanto a gravação de dados em uma tabela do DynamoDB de outra conta da AWS. O AWS Glue também suporta leitura de uma tabela do

DynamoDB em outra região e gravação em uma tabela do DynamoDB em outra região. Esta seção fornece instruções sobre como configurar o acesso e fornece um script de exemplo.

Os procedimentos nesta seção fazem referência a um tutorial do IAM para criar uma função do IAM e conceder acesso à função. O tutorial também discute sobre como assumir uma função, mas aqui, em vez disso, você usará um script de trabalho para assumir a função no AWS Glue. Este tutorial também contém informações sobre práticas gerais entre contas. Para ver um exemplo, consulte o Tutorial: Delegar acesso entre contas da AWS usando funções do IAM no Manual do usuário do IAM.

Criar um perfil

Siga o passo 1 no tutorial para criar uma função do IAM na conta A. Ao definir as permissões da função, você pode optar por anexar políticas existentes, como `AmazonDynamoDBReadOnlyAccess` ou `AmazonDynamoDBFullAccess`, para permitir que a função leia/grave do DynamoDB. O exemplo a seguir mostra a criação de uma função chamada `DynamoDBCrossAccessRole` com a política de permissão `AmazonDynamoDBFullAccess`.

Conceder acesso ao perfil

Siga o passo 2 no tutorial no Guia do usuário do IAM para permitir que a conta B alterne para a função recém-criada. O exemplo a seguir cria uma nova política com a seguinte instrução:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "<DynamoDBCrossAccessRole's ARN>"
    }
}
```

Em seguida, você pode anexar essa política ao grupo/função/usuário que deseja usar para acessar o DynamoDB.

Assumir o perfil no script de trabalho do AWS Glue

Agora, você pode fazer login na conta B e criar um trabalho do AWS Glue. Para criar um trabalho, consulte as instruções em Configurando as propriedades da tarefa para tarefas do Spark no AWS Glue.

No script de trabalho, você precisa usar o parâmetro `dynamodb.sts.roleArn` para assumir a função `DynamoDBCrossAccessRole`. Assumir essa função permite que você obtenha as

credenciais temporárias, que precisam ser usadas para acessar o DynamoDB na conta B. Revise estes scripts de exemplo.

Para uma leitura entre contas entre regiões (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 uma leitura entre contas entre regiões (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 uma leitura e gravação entre contas entre regiões:

```
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()
```

Conexões do Kinesis

Você pode ler e gravar o Amazon Kinesis Data Streams usando informações armazenadas em uma tabela do Data Catalog ou fornecendo informações para acessar diretamente o fluxo de dados. Você pode ler informações do Kinesis em um Spark DataFrame e depois convertê-las em um Glue. AWS DynamicFrame Você pode DynamicFrames gravar no Kinesis em um formato JSON. Se você acessar diretamente o fluxo de dados, use essas opções para fornecer as informações sobre como acessar o fluxo de dados.

Se você usar `getCatalogSource` ou `create_data_frame_from_catalog` para consumir registros de uma fonte de transmissão do Kinesis, o trabalho tem o banco de dados do catálogo de dados e as informações de nome da tabela, e pode usá-los para obter alguns parâmetros básicos para leitura da fonte de transmissão do Kinesis. Se você usar `getSource`, `getSourceWithFormat`, `createDataFrameFromOptions` ou `create_data_frame_from_options`, será necessário especificar esses parâmetros básicos usando as opções de conexão descritas aqui.

Você pode especificar as opções de conexão para o Kinesis usando os seguintes argumentos para os métodos especificados na classe `GlueContext`.

- Scala
  - `connectionOptions`: usar com `getSource`, `createDataFrameFromOptions`, `getSink`
  - `additionalOptions`: usar com `getCatalogSource`, `getCatalogSink`
  - `options`: usar com `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: usar com `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: usar com `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: usar com `getSource`, `getSink`

Para notas e restrições sobre trabalhos de ETL de streaming, consulte the section called “Notas e restrições sobre ETL de transmissão”.

Configurar o Kinesis

Para se conectar a um stream de dados do Kinesis em uma tarefa do AWS Glue Spark, você precisará de alguns pré-requisitos:

- Se estiver lendo, o trabalho AWS Glue deve ter permissões IAM de nível de acesso de leitura ao stream de dados do Kinesis.
- Se estiver gravando, o trabalho do AWS Glue deve ter permissões IAM no nível de acesso Write ao stream de dados do Kinesis.

Em certos casos, você precisará configurar pré-requisitos adicionais:

- Se sua tarefa do AWS Glue estiver configurada com conexões de rede adicionais (normalmente para se conectar a outros conjuntos de dados) e uma dessas conexões fornecer opções de rede Amazon VPC, isso direcionará seu trabalho para se comunicar pela Amazon VPC. Nesse caso, você também precisará configurar o fluxo de dados do Kinesis para se comunicar pela Amazon VPC. É possível fazer isso criando um endpoint da VPC de interface entre a Amazon VPC e o fluxo de dados do Kinesis. Para obter mais informações, consulte Using Kinesis Data Streams with Interface VPC Endpoints.
- Ao especificar Amazon Kinesis Data Streams em outra conta, você deve configurar os perfis e políticas para permitir o acesso entre contas. Para obter mais informações, consulte Exemplo: Ler de uma transmissão do Kinesis em outra conta.

Para obter mais informações sobre pré-requisitos de trabalho de ETL de streaming, consulte the section called “Trabalhos de transmissão de ETL”.

Exemplo: ler de fluxos do Kinesis

Exemplo: ler de fluxos do Kinesis

Usado em conjunto com the section called “forEachBatch”.

Exemplo para fonte de transmissão do 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)
```

Exemplo: gravação em streams do Kinesis

Exemplo: ler de fluxos do Kinesis

Usado em conjunto com the section called “forEachBatch”.

Exemplo para fonte de transmissão do 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)
```

Referência de opções de conexão do Kinesis

Designa opções de conexão para o Amazon Kinesis Data Streams.

Use as seguintes opções de conexão para fontes de dados de transmissão do Kinesis:

- `"streamARN"` (obrigatório) usado para leitura/gravação. O ARN do fluxo de dados do Kinesis.
- `"classification"` (Obrigatório para leitura) Usado para leitura. O formato de arquivo usado pelos dados no registro. Obrigatório, a menos que fornecido por meio do catálogo de dados.
- `"streamName"` (opcional) usado para leitura. O nome de um fluxo de dados do Kinesis de onde ler. Usado com `endpointUrl`.
- `"endpointUrl"` (opcional) usado para leitura. Padrão: “https://kinesis.us-east-1.amazonaws.com” O AWS endpoint do stream do Kinesis. Você não precisa alterar isso, a menos que esteja se conectando a uma região especial.
- `"partitionKey"` (opcional) usado para gravação. A chave de partição do Kinesis usada na produção de registros.
- `"delimiter"` (opcional) usado para leitura. O separador de valores usado quando a `classification` é CSV. O padrão é `","`.
- `"startingPosition"`: (opcional) usado para leitura. A posição inicial no fluxo de dados do Kinesis de onde ler os dados. Os valores possíveis são `"latest"`, `"trim_horizon"`, `"earliest"` ou uma string de timestamp no formato UTC no padrão `yyyy-mm-`

ddTHH:MM:SSZ (onde Z representa um desvio do fuso horário UTC com +/-). Por exemplo: "2023-04-04T08:00:00-04:00"). O valor padrão é "latest". Observação: a string Timestamp no formato UTC para "startingPosition" é compatível somente com a versão 4.0 ou posterior do AWS Glue.

- "failOnDataLoss": (Opcional) Falha na tarefa se algum fragmento ativo estiver ausente ou expirado. O valor padrão é "false".
- "awsSTSRoleARN": (opcional) usado para leitura/gravação. O Amazon Resource Name (ARN) da função a ser assumida usando AWS Security Token Service ()AWS STS. Essa função deve ter permissões para descrever ou ler operações de registro para o fluxo de dados do Kinesis. Você deve usar esse parâmetro ao acessar um fluxo de dados em uma conta diferente. Usado em conjunto com "awsSTSSessionName".
- "awsSTSSessionName": (opcional) usado para leitura/gravação. Um identificador para a sessão que assume a função usando o AWS STS. Você deve usar esse parâmetro ao acessar um fluxo de dados em uma conta diferente. Usado em conjunto com "awsSTSRoleARN".
- "awsSTSEndpoint": (Opcional) O AWS STS endpoint a ser usado ao se conectar ao Kinesis com uma função assumida. Isso permite usar o AWS STS endpoint regional em uma VPC, o que não é possível com o endpoint global padrão.
- "maxFetchTimeInMs": (opcional) usado para leitura. O tempo máximo para o executor do trabalho ler registros referentes ao lote atual do fluxo de dados do Kinesis especificado em milissegundos (ms). Várias chamadas de API GetRecords podem ser feitas nesse período. O valor padrão é 1000.
- "maxFetchRecordsPerShard": (opcional) usado para leitura. O número máximo de registros a serem obtidos por fragmento no fluxo de dados do Kinesis por microlote. Observação: o cliente poderá exceder esse limite se o trabalho de streaming já tiver lido registros extras do Kinesis (na mesma chamada get-records). Se maxFetchRecordsPerShard precisa ser rigoroso, então precisa ser um múltiplo de maxRecordPerRead. O valor padrão é 100000.
- "maxRecordPerRead": (opcional) usado para leitura. O número máximo de registros a serem obtidos por fragmento no fluxo de dados do Kinesis em cada operação getRecords. O valor padrão é 10000.
- "addIdleTimeBetweenReads": (opcional) usado para leitura. Adiciona um atraso de tempo entre duas operações getRecords. O valor padrão é "False". Essa opção só pode ser configurada para o Glue versão 2.0 e posterior.

- `"idleTimeBetweenReadsInMs"`: (opcional) usado para leitura. O atraso mínimo entre duas operações , especificado em ms. O valor padrão é `1000`. Essa opção só pode ser configurada para o Glue versão 2.0 e posterior.
- `"describeShardInterval"`: (opcional) usado para leitura. O intervalo de tempo mínimo entre duas chamadas de API `ListShards` para que seu script considere a refragmentação. Para obter mais informações, consulte Estratégias para refragmentação no Guia do desenvolvedor do Amazon Kinesis Data Streams. O valor padrão é `1s`.
- `"numRetries"`: (opcional) usado para leitura. O número máximo de novas tentativas para solicitações de API do Kinesis Data Streams. O valor padrão é `3`.
- `"retryIntervalMs"`: (opcional) usado para leitura. O período de espera (especificado em ms) antes de repetir a chamada da API Kinesis Data Streams. O valor padrão é `1000`.
- `"maxRetryIntervalMs"`: (opcional) usado para leitura. O período de espera máximo (especificado em ms) entre duas tentativas de uma chamada de API Kinesis Data Streams. O valor padrão é `10000`.
- `"avoidEmptyBatches"`: (opcional) usado para leitura. Evita a criação de um trabalho de micro lote vazio verificando se há dados não lidos no fluxo de dados do Kinesis antes de o lote ser iniciado. O valor padrão é `"False"`.
- `"schema"`: (Obrigatório quando inferSchema é definido como false): usado para leitura. O esquema a ser usado para processar a carga. Se a classificação for `avro`, o esquema fornecido deverá estar no formato de esquema Avro. Se a classificação não for `avro`, o esquema fornecido deverá estar no formato de esquema DDL.

  Veja a seguir alguns exemplos de esquema.

  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) usado para leitura. O valor padrão é "false". Se definido como “true”, o esquema será detectado em runtime com base na carga útil em `foreachbatch`.
- `"avroSchema"`: (Obsoleto) Usado para leitura. Parâmetro usado para especificar um esquema de dados Avro quando o formato Avro é usado. Esse parâmetro foi descontinuado. Use o parâmetro `schema`.
- `"addRecordTimestamp"`: (opcional) usado para leitura. Quando essa opção for definida como "true", a saída de dados conterá uma coluna adicional denominada "__src_timestamp" que indica a hora que o registro correspondente é recebido pelo fluxo. O valor padrão é "false". Essa opção é compatível com o AWS Glue versão 4.0 ou posterior.
- `"emitConsumerLagMetrics"`: (opcional) usado para leitura. Quando a opção é definida como “verdadeira”, para cada lote, ela emitirá as métricas da duração entre o registro mais antigo recebido pelo stream e o horário em AWS Glue que ele chega. CloudWatch O nome da métrica é “glue.driver.streaming”. maxConsumerLagInMs”. O valor padrão é "false". Essa opção é compatível com o AWS Glue versão 4.0 ou posterior.
- `"fanoutConsumerARN"`: (opcional) usado para leitura. O ARN de um consumidor de fluxo do Kinesis para o fluxo especificado em `streamARN`. Usado para habilitar o modo de distribuição avançada para a conexão do Kinesis. Para obter mais informações sobre como consumir um fluxo do Kinesis com distribuição avançada, consulte the section called “Usar distribuição avançada nas tarefas de streaming do Kinesis”.
- `"recordMaxBufferedTime"` (opcional) usado para gravação. Padrão: 1000 (ms). Tempo máximo em que um registro é armazenado em buffer enquanto espera para ser gravado.

- `"aggregationEnabled"` (opcional) usado para gravação. Padrão: true. Especifica se os registros devem ser agregados antes de serem enviados para o Kinesis.
- `"aggregationMaxSize"` (opcional) usado para gravação. Padrão: 51200 (bytes) Se um registro for maior que esse limite, ele ignorará o agregador. Nota: O Kinesis impõe um limite de 50 KB no tamanho do registro. Se você definir isso além de 50 KB, registros grandes serão rejeitados pelo Kinesis.
- `"aggregationMaxCount"` (opcional) usado para gravação. Padrão: 4294967295. O número máximo de itens a serem retornados em um registro agregado.
- `"producerRateLimit"` (opcional) usado para gravação. Padrão: 150 (%). Limita o throughput por fragmento enviado por um único produtor (como seu trabalho), como uma porcentagem do limite de back-end.
- `"collectionMaxCount"` (opcional) usado para gravação. Padrão: 500. Número máximo de itens a serem embalados em uma PutRecords solicitação.
- `"collectionMaxSize"` (opcional) usado para gravação. Padrão: 5242880 (bytes) Quantidade máxima de dados a serem enviados com uma PutRecords solicitação.

Usar distribuição avançada nas tarefas de streaming do Kinesis

Um consumidor de distribuição avançada pode receber registros de um fluxo do Kinesis com um throughput dedicado que pode ser maior que o dos consumidores comuns. Isso é feito otimizando o protocolo de transferência usado para fornecer dados a um consumidor do Kinesis, como o seu trabalho. Para obter mais informações sobre distribuição avançada do Kinesis, consulte a documentação do Kinesis.

No modo de distribuição avançada, as opções de conexão `maxRecordPerRead` e `idleTimeBetweenReadsInMs` não se aplicam mais, pois esses parâmetros não são configuráveis quando se usa o fan-out aprimorado. As opções de configuração para novas tentativas funcionam conforme descrito.

Use os procedimentos a seguir para habilitar e desabilitar a distribuição avançada para seu trabalho de streaming. Você deve registrar um consumidor de fluxo para cada trabalho que consuma dados do fluxo.

Para permitir um maior consumo de distribuição avançada no trabalho:

1. Registre um consumidor de fluxo para o trabalho usando a API do Kinesis. Siga as instruções para register a consumer with enhanced fan-out using the Kinesis Data Streams

API na documentação do Kinesis. Você só precisará seguir a primeira etapa: chamar RegisterStreamConsumer. Sua solicitação deve retornar um ARN, *consumerARN*.

2. Defina a opção de conexão `fanoutConsumerARN` como *consumerARN* nos argumentos do método de conexão.
3. Reinicie seu trabalho.

Para desabilitar o consumo de distribuição avançada no trabalho:

1. Remova a opção de conexão `fanoutConsumerARN` da sua chamada de método.
2. Reinicie seu trabalho.
3. Siga as instruções para deregister a consumer na documentação do Kinesis. Essas instruções se aplicam ao console, mas também podem ser obtidas por meio da API Kinesis. Para obter mais informações sobre o cancelamento do registro de consumidores de fluxo por meio da API do Kinesis, consulte DeregisterStreamConsumer na documentação do Kinesis.

Conexões do Amazon S3

Você pode usar o AWS Glue para Spark para ler e gravar arquivos no Amazon S3. O AWS Glue para Spark é compatível com muitos formatos de dados comuns armazenados no Amazon S3 prontos para uso, incluindo CSV, Avro, JSON, Orc e Parquet. Para obter mais informações sobre os formatos de dados compatíveis, consulte the section called “Opções de formato de dados”. Cada formato de dados pode ser compatível um conjunto diferente de atributos do AWS Glue. Consulte a página do seu formato de dados para obter os detalhes específicos da compatibilidade com atributos. Além disso, você pode ler e gravar arquivos versionados armazenados nas estruturas de data lake do Hudi, Iceberg e do Delta Lake. Para obter mais informações sobre estruturas de data lake, consulte the section called “Estruturas de data lake”.

Com o AWS Glue, você pode particionar os objetos do Amazon S3 em uma estrutura de pastas durante a gravação e depois recuperá-los por partição para melhorar a performance usando uma configuração simples. Você também pode definir a configuração para agrupar arquivos pequenos ao transformar seus dados, para melhorar a performance. Você pode ler e gravar arquivos `bzip2` e `gzip` no Amazon S3.

Tópicos

- Configurar conexões do S3
- Referência de opções de conexão do Amazon S3

- Sintaxes de conexão obsoletas para formatos de dados
- Excluir classes de armazenamento do Amazon S3
- Gerenciar partições para saída de ETL no AWS Glue
- Ler arquivos de entrada em grupos maiores
- Endpoints da Amazon VPC para o Amazon S3

Configurar conexões do S3

Para se conectar ao Amazon S3 em um trabalho do AWS Glue com Spark, você precisará de alguns pré-requisitos:

- O trabalho do AWS Glue deve ter permissões do IAM para buckets relevantes do Amazon S3.

Em certos casos, você precisará configurar pré-requisitos adicionais:

- Ao configurar o acesso entre contas, os controles de acesso apropriados no bucket do Amazon S3.
- Por motivos de segurança, você pode optar por rotear suas solicitações do Amazon S3 por meio de uma Amazon VPC. Essa abordagem pode introduzir desafios de largura de banda e disponibilidade. Para ter mais informações, consulte the section called “Endpoints da Amazon VPC para o Amazon S3”.

Referência de opções de conexão do Amazon S3

Designa uma conexão com o Amazon S3.

Como o Amazon S3 gerencia arquivos em vez de tabelas, além de especificar as propriedades de conexão fornecidas neste documento, você precisará especificar configurações adicionais sobre seu tipo de arquivo. Você especifica essas informações por meio de opções de formato de dados. Para obter mais informações sobre essas opções de formato, consulte the section called “Opções de formato de dados”. Você também pode especificar essas informações fazendo a integração com o catálogo de dados do AWS Glue.

Para obter um exemplo da distinção entre opções de conexão e opções de formato, considere como o método the section called “create_dynamic_frame_from_options” usa `connection_type`, `connection_options`, `format` e `format_options`. Esta seção discute especificamente os parâmetros fornecidos às `connection_options`.

Use as seguintes opções de conexão com `"connectionType": "s3"`:

- `"paths"`: (obrigatório) uma lista de caminhos do Amazon S3 dos quais fazer a leitura.
- `"exclusions"`: (Opcional) Uma string contendo uma lista JSON de padrões glob de estilo Unix padrões para excluir. Por exemplo, `"[\"**.pdf\"]"` exclui todos os arquivos PDF. Para obter mais informações sobre a sintaxe glob compatível com o AWS Glue, consulte Incluir e excluir padrões.
- `"compressionType"`: ou `"compression"`: (opcional) especifica como os dados são compactados. Use `"compressionType"` para fontes do Amazon S3 e `"compression"` para destinos do Amazon S3. Isso geralmente não é necessário se os dados tem uma extensão de arquivo padrão. Os possíveis valores são `"gzip"` e `"bzip2"`). Formatos de compactação adicionais podem ser compatíveis com formatos específicos. Para obter os detalhes específicos da compatibilidade com atributos, consulte a página do seu formato de dados.
- `"groupFiles"`: (opcional) o agrupamento de arquivos é ativado por padrão quando a entrada contiver mais de 50.000 arquivos. Para habilitar o agrupamento com menos de 50.000 arquivos, defina esse parâmetro como `"inPartition"`. Para desabilitar o agrupamento quando houver mais de 50.000 arquivos, defina esse parâmetro como `"none"`.
- `"groupSize"`: (Opcional) O tamanho do grupo de destino em bytes. O padrão é calculado com base no tamanho de dados de entrada e o tamanho de seu cluster. Quando há menos de 50.000 arquivos de entrada, `"groupFiles"` deve ser definido como `"inPartition"` para poder entrar em vigor.
- `"recurse"`: (Opcional) Se definido como verdadeiro, recursivamente lê arquivos em todos os subdiretórios de acordo com os caminhos especificados.
- `"maxBand"`: (opcional, avançado) essa opção controla a duração, em milissegundos, após a qual a listagem `s3` provavelmente será consistente. Os arquivos com carimbos de data e hora de modificação que estão dentro dos últimos `maxBand` milissegundos são rastreados principalmente ao usar `JobBookmarks` para considerar a consistência final do Amazon S3. A maioria dos usuários não precisa definir essa opção. O valor padrão é 900.000 milissegundos, ou 15 minutos.
- `"maxFilesInBand"`: (opcional, avançado) essa opção especifica o número máximo de arquivos que devem ser salvos dos últimos `maxBand` segundos. Se esse número for excedido, os arquivos extras são ignorados e apenas processados na próxima execução do trabalho. A maioria dos usuários não precisa definir essa opção.
- `"isFailFast"`: (opcional) essa opção determina se um trabalho de ETL do AWS Glue lança exceções de análise do leitor. Se definido como `true`, os trabalhos falham rapidamente se quatro tentativas da tarefa do Spark falharem em analisar os dados corretamente.

- `"catalogPartitionPredicate"`: (opcional) usado para leitura. O conteúdo de uma cláusula SQL `WHERE`. Usado ao ler tabelas do catálogo de dados com uma quantidade muito grande de partições. Recupera partições correspondentes dos índices do catálogo de dados. Usado com `push_down_predicate`, uma opção do método the section called “create_dynamic_frame_from_catalog” (e outros métodos similares). Para ter mais informações, consulte the section called “Predicados de partição de catálogo”.
- `"partitionKeys"`: (opcional) usado para gravação. Uma matriz de strings de rótulo da coluna. O AWS Glue particionará os dados conforme especificado por essa configuração. Para ter mais informações, consulte the section called “Gravar partições”.
- `"excludeStorageClasses"`: (opcional) usado para leitura. Uma matriz de strings especificando as classes de armazenamento do Amazon S3. O AWS Glue excluirá objetos do Amazon S3 com base nessa configuração. Para ter mais informações, consulte the section called “Excluir classes de armazenamento do Amazon S3”.

Sintaxes de conexão obsoletas para formatos de dados

Certos formatos de dados podem ser acessados usando a sintaxe de um tipo de conexão específico. Essa sintaxe está obsoleta. Em seu lugar, recomendamos que você especifique os formatos usando o tipo de conexão `s3` e as opções de formato fornecidas em the section called “Opções de formato de dados”.

“connectionType”: “orc”

Designa uma conexão para arquivos armazenados no Amazon S3 no formato de arquivo Apache Hive Optimized Row Columnar (ORC).

Use as seguintes opções de conexão com `"connectionType": "orc"`:

- `paths`: (obrigatório) uma lista de caminhos do Amazon S3 dos quais fazer a leitura.
- (Outras opções de pares de nome/valor): qualquer opção adicional, incluindo opções de formatação, são transmitidas diretamente à `DataSource` do SparkSQL.

"connectionType": "parquet"

Designa uma conexão com arquivos armazenados no Amazon S3 no formato de arquivo Apache Parquet.

Use as seguintes opções de conexão com `"connectionType": "parquet"`:

- `paths`: (obrigatório) uma lista de caminhos do Amazon S3 dos quais fazer a leitura.
- (Outras opções de pares de nome/valor): qualquer opção adicional, incluindo opções de formatação, são transmitidas diretamente à `DataSource` do SparkSQL.

Excluir classes de armazenamento do Amazon S3

Se você estiver executando trabalhos de ETL do AWS Glue que leiam arquivos ou partições do Amazon Simple Storage Service (Amazon S3), poderá excluir alguns tipos de classe de armazenamento do Amazon S3.

As seguintes classes de armazenamento estão disponíveis no Amazon S3:

- `STANDARD`: para armazenamento de uso geral de dados acessados com frequência.
- `INTELLIGENT_TIERING`: para dados com padrões de acesso desconhecidos ou inconstantes.
- `STANDARD_IA` e `ONEZONE_IA`: para dados de longa duração, mas acessados com menos frequência.
- `GLACIER`, `DEEP_ARCHIVE` e `REDUCED_REDUNDANCY`: para arquivamento de longo prazo e preservação digital.

Para obter mais informações, consulte Classes de armazenamento do Amazon S3 no Guia do desenvolvedor do Amazon S3.

Os exemplos nesta seção mostram como excluir as classes de armazenamento `GLACIER` e `DEEP_ARCHIVE`. Essas classes permitem que você liste arquivos, mas elas não permitem que você leia os arquivos, a menos que sejam restaurados. (Para obter mais informações, consulte Restaurar objetos arquivados no Guia do desenvolvedor do Amazon S3).

Ao usar exclusões de classe de armazenamento, você pode garantir que seus trabalhos do AWS Glue funcionarão em tabelas com partições entre esses níveis de classe de armazenamento. Sem exclusões, os trabalhos que leem dados desses níveis falham com o seguinte erro: AmazonS3Exception: The operation is not valid for the object's storage class (AmazonS3Exception: a operação não é válida para a classe de armazenamento do objeto).

Há diferentes maneiras de filtrar classes de armazenamento do Amazon S3 no AWS Glue.

Tópicos

- Excluir classes de armazenamento do Amazon S3 ao criar um quadro dinâmico

- Excluir classes de armazenamento do Amazon S3 em uma tabela do Data Catalog

Excluir classes de armazenamento do Amazon S3 ao criar um quadro dinâmico

Para excluir classes de armazenamento do Amazon S3 ao criar um quadro dinâmico, use `excludeStorageClasses` em `additionalOptions`. O AWS Glue usa automaticamente sua própria implementação do `Lister` do Amazon S3 para listar e excluir arquivos correspondentes às classes de armazenamento especificadas.

Os exemplos de Python e Scala a seguir mostram como excluir as classes de armazenamento `GLACIER` e `DEEP_ARCHIVE` ao criar um quadro dinâmico.

Exemplo do 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"]
    }
)
```

Exemplo do Scala:

```
val* *df = glueContext.getCatalogSource(
    nameSpace, tableName, "", "my_transformation_context",
    additionalOptions = JsonOptions(
        Map("excludeStorageClasses" -> List("GLACIER", "DEEP_ARCHIVE"))
    )
).getDynamicFrame()
```

Excluir classes de armazenamento do Amazon S3 em uma tabela do Data Catalog

É possível especificar exclusões de classe de armazenamento a serem usadas por um trabalho de ETL do AWS Glue como um parâmetro de tabela no catálogo de dados do Glue da AWS. Você pode incluir esse parâmetro na operação `CreateTable` usando a AWS Command Line Interface (AWS CLI) ou de forma programática com a API. Para obter mais informações, consulte Estrutura de tabelas e CreateTable.

Você também pode especificar classes de armazenamento excluídas no console do AWS Glue.

Como excluir classes de armazenamento do Amazon S3 (console)

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação à esquerda, escolha Tables (Tabelas).
3. Escolha o nome da tabela na lista e, depois, escolha Edit table (Editar tabela).
4. Em Table properties (Propriedades da tabela), adicione **`excludeStorageClasses`** como uma chave e **`[\"GLACIER\",\"DEEP_ARCHIVE\"]`** como um valor.
5. Escolha Aplicar.

Gerenciar partições para saída de ETL no AWS Glue

O particionamento é uma técnica importante para organizar os conjuntos de dados para que possam ser consultados com eficiência. Ele organiza dados em uma estrutura hierárquica de diretórios com base nos valores distintos de uma ou mais colunas.

Por exemplo, você pode decidir particionar os logs da sua aplicação no Amazon Simple Storage Service (Amazon S3) por data, divididos em ano, mês e dia. Os arquivos que correspondem a um único dia de dados são colocados com um prefixo, como `s3://my_bucket/logs/year=2018/month=01/day=23/`. Sistemas como o Amazon Athena, o Amazon Redshift Spectrum e agora o AWS Glue podem usar essas partições para filtrar dados por valor de partição sem precisar ler todos os dados subjacentes do Amazon S3.

Os crawlers não apenas inferem tipos de arquivo e esquemas, como também identificam automaticamente a estrutura da partição do seu conjunto de dados quando eles preenchem o AWS Glue Data Catalog. As colunas resultantes da partição ficam disponíveis para consulta em trabalhos de ETL do AWS Glue ou em mecanismos de consulta, como o Amazon Athena.

Após rastrear uma tabela, você pode visualizar as partições que o crawler criou. No console do AWS Glue, escolha Tables (Tabelas) no painel de navegação esquerdo. Escolha a tabela criada pelo crawler e, em seguida, selecione View Partitions (Exibir partições).

Para os caminhos particionados do Apache Hive no estilo `key=val`, os crawlers preenchem automaticamente o nome da coluna usando o nome da chave padrão. Caso contrário, ele usa nomes padrão, como `partition_0`, `partition_1` e assim por diante. Você pode alterar os

nomes padrão no console. Para fazer isso, navegue até a tabela. Verifique se os índices existem na guia Índices. Se for esse o caso, você precisa excluí-las para continuar (você poderá recriá-las posteriormente usando os novos nomes das colunas). Em seguida, escolha Editar esquema e modifique os nomes das colunas de partição.

Nos scripts de ETL, você pode filtrar as colunas da partição. Uma vez que as informações de partição são armazenadas no Data Catalog, utilize as chamadas de API `from_catalog` para incluir as colunas de partição no `DynamicFrame`. Por exemplo, use `create_dynamic_frame.from_catalog` em vez de `create_dynamic_frame.from_options`.

Particionamento é uma técnica de otimização que reduz as varreduras de dados. Para obter mais informações sobre o processo para identificar quando essa técnica é adequada, consulte Reduce the amount of data scan no guia Best practices for performance tuning AWS Glue for Apache Spark jobs em AWS Prescriptive Guidance.

Pré-filtragem usando a aplicação de predicados

Em muitos casos, você pode usar uma aplicação de predicado para filtrar partições sem precisar listar e ler todos os arquivos do seu conjunto de dados. Em vez de ler todo o conjunto de dados e, em seguida, filtrá-lo em um DynamicFrame, você pode aplicar o filtro diretamente nos metadados da partição no Data Catalog. Em seguida, você lista e lê somente o que você realmente precisa em um DynamicFrame.

Por exemplo, em Python, você pode gravar o seguinte.

```
glue_context.create_dynamic_frame.from_catalog(
    database = "my_S3_data_set",
    table_name = "catalog_data_table",
    push_down_predicate = my_partition_predicate)
```

Isso cria um DynamicFrame que carrega somente as partições do Data Catalog que satisfazem a expressão do predicado. Dependendo do tamanho do subconjunto dos dados que você está carregando, isso pode economizar muito tempo de processamento.

A expressão do predicado pode ser qualquer expressão booleana compatível com Spark SQL. Tudo que você pode colocar em uma cláusula `WHERE` de uma consulta do SQL Spark funcionará. Por exemplo, a expressão de predicado `pushDownPredicate = "(year=='2017' and month=='04')"` carrega apenas as partições no Data Catalog que tiverem tanto `year` igual a 2017

quanto `month` igual a 04. Para obter mais informações, consulte a Documentação do Apache Spark SQL e, em especial, a Referência das funções SQL em Scala.

Filtragem do lado do servidor usando predicados de partição de catálogo

A opção `push_down_predicate` é aplicada depois de listar todas as partições do catálogo e antes de listar arquivos do Amazon S3 para essas partições. Se você tiver muitas partições para uma tabela, a listagem de partições de catálogo ainda poderá incorrer em sobrecarga de tempo adicional. Para resolver essa sobrecarga, você pode usar a remoção de partição do lado do servidor com a opção `catalogPartitionPredicate`, que usaíndices de partição no AWS Glue Data Catalog. Isso torna a filtragem de partição muito mais rápida quando você tem milhões de partições em uma tabela. Você pode usar `push_down_predicate` e `catalogPartitionPredicate` em `additional_options` juntos, se seu `catalogPartitionPredicate` exigir sintaxe de predicado que ainda não é suportada com os índices de partição de 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` e `catalogPartitionPredicate` usam sintaxes diferentes. O primeiro usa a sintaxe padrão do Spark SQL e o outro usa o analisador JSQL.

Gravar partições

Por padrão, um DynamicFrame não é particionado quando é gravado. Todos os arquivos de saída são gravados no nível superior do caminho de saída especificado. Até recentemente, a única maneira de gravar um DynamicFrame em partições era convertê-lo em um DataFrame Spark SQL antes da gravação.

No entanto, DynamicFrames agora oferecem suporte ao particionamento nativo usando uma sequência de chaves, usando a opção `partitionKeys` ao criar um depósito. Por exemplo, o seguinte código Python grava um conjunto de dados no Amazon S3 no formato Parquet em diretórios particionados, de acordo com o tipo de campo. A partir daí, você pode processar essas partições usando outros sistemas, como o Amazon Athena.

```
glue_context.write_dynamic_frame.from_options(
    frame = projectedEvents,
    connection_type = "s3",
    connection_options = {"path": "$outpath", "partitionKeys": ["type"]},
    format = "parquet")
```

Ler arquivos de entrada em grupos maiores

Você pode definir as propriedades das tabelas para habilitar um trabalho de ETL do AWS Glue a fim de agrupar arquivos quando eles são lidos em um armazenamento de dados do Amazon S3. Essas propriedades permitem que cada tarefa de ETL leia um grupo de arquivos de entrada em uma única partição na memória. Isso é especialmente útil quando há um grande número de arquivos pequenos no armazenamento de dados do Amazon S3. Ao definir determinadas propriedades, você instrui o AWS Glue a agrupar arquivos em uma partição de dados do Amazon S3 e a definir o tamanho dos grupos a serem lidos. Você também pode definir essas opções durante a leitura em um armazenamento de dados do Amazon S3 com o método `create_dynamic_frame.from_options`.

Para habilitar o agrupamento de arquivos para uma tabela, defina pares de chave/valor no campo de parâmetros da estrutura da tabela. Use a notação JSON para definir um valor para o campo de parâmetro da tabela. Para obter mais informações sobre como editar as propriedades de uma tabela, consulte Exibir e editar detalhes da tabela.

Você pode usar esse método para habilitar o agrupamento de tabelas no Data Catalog com armazenamentos de dados do Amazon S3.

groupFiles

Defina groupFiles (agrupar arquivos) como `inPartition` para habilitar o agrupamento de arquivos em uma partição de dados do Amazon S3. O AWS Glue habilitará automaticamente o agrupamento se houver mais de 50.000 arquivos de entrada, como no exemplo a seguir.

```
'groupFiles': 'inPartition'
```

groupSize

Defina groupSize como o tamanho de destino de grupos em bytes. A propriedade groupSize é opcional. Se não for fornecida, o AWS Glue calculará um tamanho para usar todos os núcleos de CPU no cluster e, ao mesmo tempo, ainda reduzindo o número total de tarefas de ETL e partições na memória.

Por exemplo, o seguinte define o tamanho do grupo como 1 MB.

```
'groupSize': '1048576'
```

Observe que o `groupsize` deve ser definido com o resultado de um cálculo. Por exemplo 1024 * 1024 = 1048576.

recurse

Defina recursivo para `True` recursivamente ler arquivos em todos os subdiretórios ao especificar `paths` uma matriz de caminhos. Você não precisará definir recurse (recursivo) se `paths` for uma matriz de chaves de objeto no Amazon S3 ou se o formato de entrada for parquet/orc, como no exemplo a seguir.

```
'recurse':True
```

Se você estiver lendo no Amazon S3 diretamente usando o método `create_dynamic_frame.from_options`, adicione estas opções de conexão. Por exemplo, as seguintes tentativas de agrupar arquivos em grupos de 1 MB.

```
df = glueContext.create_dynamic_frame.from_options("s3", {'paths': ["s3://s3path/"],
 'recurse':True, 'groupFiles': 'inPartition', 'groupSize': '1048576'}, format="json")
```

> **Note**
>
> Há suporte para `groupFiles` para DynamicFrames criados a partir dos seguintes formatos de dados: csv, ion, grokLog, json e xml. Não há suporte dessa opção para avro, parquet e orc.

Endpoints da Amazon VPC para o Amazon S3

Por motivos de segurança, muitos clientes da AWS executam suas aplicações em um ambiente da Amazon Virtual Private Cloud (Amazon VPC). Com a Amazon VPC, você pode executar instâncias do Amazon EC2 em uma nuvem privada virtual, a qual é isolada logicamente de outras redes, incluindo a Internet pública. Com uma Amazon VPC, você tem controle sobre o intervalo de endereços IP, sub-redes, tabelas de roteamento, gateways de rede e configurações de segurança.

> **Note**
>
> Caso tenha criado sua conta da AWS após 4/12/2013, você já tem uma VPC padrão em cada região da AWS. Você pode começar a usar sua VPC padrão imediatamente sem qualquer configuração adicional.
> Para obter mais informações, consulte Suas sub-redes e VPC padrão no Manual do usuário da Amazon VPC.

Muitos clientes têm preocupações legítimas com a segurança e a privacidade sobre o envio e o recebimento de dados pela Internet pública. Os clientes podem resolver essas preocupações usando uma rede privada virtual (VPN) para rotear todo o tráfego de rede do Amazon S3 por meio da infraestrutura de rede corporativa deles. No entanto, essa abordagem pode introduzir desafios de largura de banda e disponibilidade.

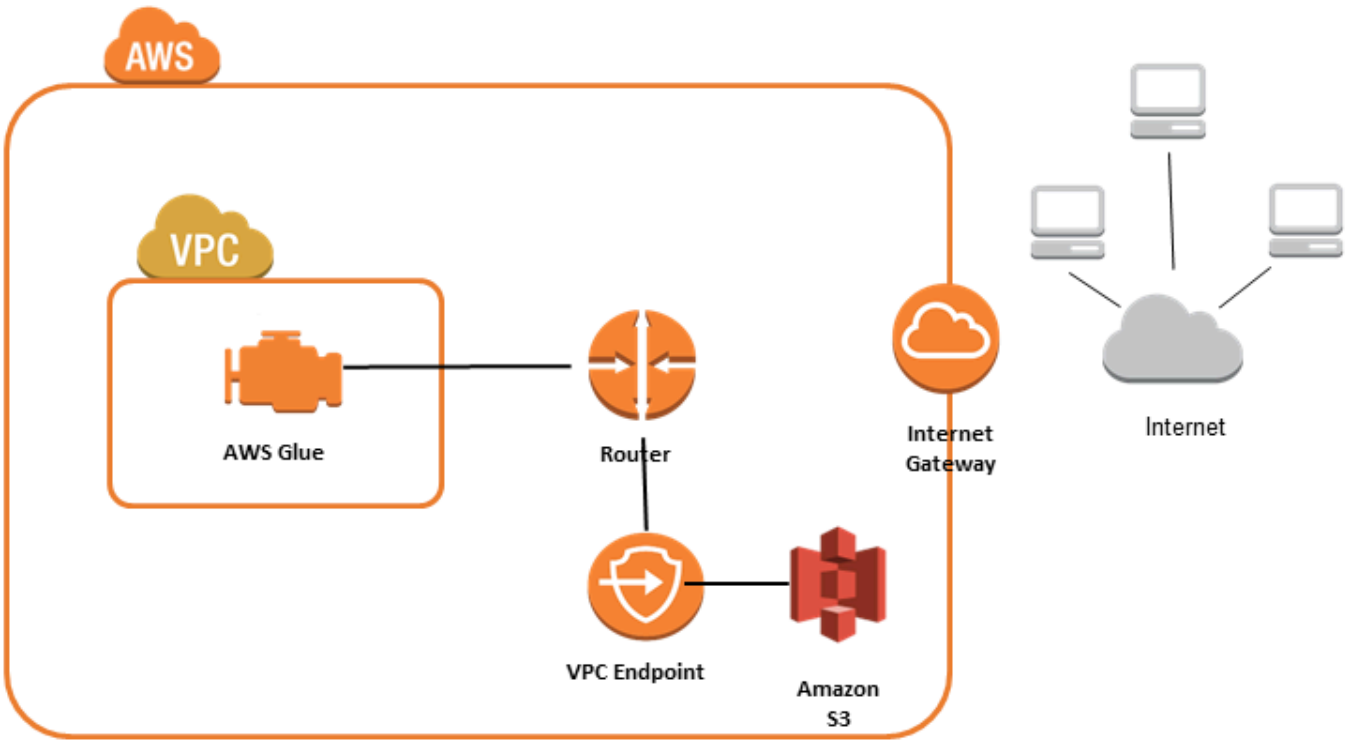
Os endpoints da VPC para o Amazon S3 podem reduzir esses desafios. Um endpoint da VPC para o Amazon S3 permite ao AWS Glue usar endereços IP privados para acessar o Amazon S3 sem exposição à Internet pública. O AWS Glue não precisa de endereços IP públicos e você não precisa de um gateway da Internet, de um dispositivo NAT ou de um gateway privado virtual na sua VPC.

Use as políticas de endpoint para controlar o acesso ao Amazon S3. O tráfego entre sua VPC e o produto da AWS não sai da rede da Amazon.

Ao criar um endpoint da VPC para o Amazon S3, quaisquer solicitações para um endpoint do Amazon S3 dentro da região (por exemplo: s3.us-west-2.amazonaws.com) são encaminhadas para um endpoint privado do Amazon S3 dentro da rede da Amazon. Você não precisa modificar suas aplicações em execução em instâncias do Amazon EC2 na VPC. O nome do endpoint permanece o mesmo, mas a rota para o Amazon S3 permanece inteiramente dentro da rede da Amazon e não acessa a Internet pública.

Para obter mais informações sobre o endpoints da VPC, consulte Endpoints da VPC no Manual do usuário da Amazon VPC.

O diagrama a seguir mostra como o AWS Glue pode usar um endpoint da VPC para acessar o Amazon S3.



Para configurar o acesso ao Amazon S3

1. Faça login no AWS Management Console e abra o console do Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No painel de navegação à esquerda, escolha Endpoints.
3. Escolha Create Endpoint (Criar endpoint) e siga as etapas para criar um endpoint da VPC do Amazon S3 do tipo Gateway.

Conexões do Amazon DocumentDB

É possível usar o AWS Glue para Spark para ler e gravar em tabelas no Amazon DocumentDB. Você pode se conectar ao Amazon DocumentDB usando credenciais armazenadas no AWS Secrets Manager por meio de uma conexão do AWS Glue.

Para obter mais informações sobre o Amazon DocumentDB, consulte a Documentação do Amazon DocumentDB.

> **Note**
>
> No momento, os clusters elásticos do Amazon DocumentDB não são compatíveis com o uso do conector do AWS Glue. Para obter mais informações sobre clusters elásticos, consulte Using Amazon DocumentDB elastic clusters.

Ler e gravar em coleções do Amazon DocumentDB

> **Note**
>
> Ao criar um trabalho de ETL que se conecta ao Amazon DocumentDB, para a propriedade de trabalho `Connections`, é necessário designar um objeto de conexão que especifica a nuvem privada virtual (VPC) em que o Amazon DocumentDB é executado. Para o objeto de conexão, o tipo de conexão deve ser `JDBC`, e o `JDBC URL` deve ser `mongo://<DocumentDB_host>:27017`.

> **Note**
>
> Estes exemplos de código foram desenvolvidos para o AWS Glue 3.0. Para migrar para o AWS Glue 4.0, consulte the section called “MongoDB”. O parâmetro `uri` foi alterado.

> **Note**
>
> Ao usar o Amazon DocumentDB, `retryWrites` deve ser definido como falso em determinadas situações, como quando o documento escrito especifica `_id`. Para obter mais informações, consulte Diferenças funcionais com o MongoDB na documentação do Amazon DocumentDB.

O script Python a seguir demonstra como usar tipos e opções de conexão para ler e gravar no 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()
```

O script Scala a seguir demonstra como usar tipos e opções de conexão para ler e gravar no 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)
  }
}
```

Referência de opções de conexão do Amazon DocumentDB

Designa uma conexão com o Amazon DocumentDB (compatível com MongoDB).

As opções de conexão diferem para uma conexão da fonte e uma conexão do coletor.

“connectionType”: “documentdb” como fonte

Use as seguintes opções de conexão com `"connectionType": "documentdb"` como fonte:

- `"uri"`: (obrigatório) o host do Amazon DocumentDB do qual fazer a leitura, formatado como `mongodb://<host>:<port>`.
- `"database"`: (obrigatório) o banco de dados do Amazon DocumentDB do qual fazer a leitura.
- `"collection"`: (obrigatório) a coleção do Amazon DocumentDB da qual fazer a leitura.
- `"username"`: (obrigatório) o nome de usuário do Amazon DocumentDB.
- `"password"`: (obrigatório) a senha do Amazon DocumentDB.
- `"ssl"`: (obrigatório se estiver usando SSL) se sua conexão usar SSL, você deve incluir essa opção com o valor `"true"`.
- `"ssl.domain_match"`: (obrigatório se estiver usando SSL) se sua conexão usar SSL, você deve incluir essa opção com o valor `"false"`.
- `"batchSize"`: (opcional) o número de documentos que devem ser retornados por lote, usado no cursor de lotes internos.
- `"partitioner"`: (opcional) o nome da classe do particionador do qual deve ser feita a leitura dos dados de entrada do Amazon DocumentDB. O conector fornece os seguintes particionadores:
  - `MongoDefaultPartitioner` (padrão) (sem suporte no AWS Glue 4.0)
  - `MongoSamplePartitioner` (sem suporte no AWS Glue 4.0)
  - `MongoShardedPartitioner`
  - `MongoSplitVectorPartitioner`
  - `MongoPaginateByCountPartitioner`
  - `MongoPaginateBySizePartitioner` (sem suporte no AWS Glue 4.0)
- `"partitionerOptions"`: (opcional) opções para o particionador designado. As seguintes opções são compatíveis com cada particionador:
  - `MongoSamplePartitioner`: `partitionKey`, `partitionSizeMB`, `samplesPerPartition`
  - `MongoShardedPartitioner`: `shardkey`
  - `MongoSplitVectorPartitioner`: `partitionKey`, partitionSizeMB
  - `MongoPaginateByCountPartitioner`: `partitionKey`, `numberOfPartitions`
  - `MongoPaginateBySizePartitioner`: `partitionKey`, partitionSizeMB

Para obter mais informações sobre essas opções, consulte Partitioner Configuration na documentação do MongoDB.

“connectionType”: “documentdb” como coletor

Use as seguintes opções de conexão com `"connectionType": "documentdb"` como coletor:

- `"uri"`: (obrigatório) o host do Amazon DocumentDB no qual fazer a gravação, formatado como `mongodb://<host>:<port>`.
- `"database"`: (obrigatório) o banco de dados do Amazon DocumentDB no qual fazer a gravação.
- `"collection"`: (obrigatório) a coleção do Amazon DocumentDB na qual fazer a gravação.
- `"username"`: (obrigatório) o nome de usuário do Amazon DocumentDB.
- `"password"`: (obrigatório) a senha do Amazon DocumentDB.
- `"extendedBsonTypes"`: (opcional) se `true`, permite tipos BSON estendidos ao gravar dados no Amazon DocumentDB. O padrão é `true`.
- `"replaceDocument"`: (opcional) se for `true`, substituirá todo o documento ao salvar conjuntos de dados que contêm um campo `_id`. Se for `false`, serão atualizados somente os campos do documento que correspondam aos campos do conjunto de dados. O padrão é `true`.
- `"maxBatchSize"`: (opcional) o tamanho máximo do lote para operações em massa ao salvar dados. O padrão é 512.
- `"retryWrites"`: (Opcional): repita automaticamente determinadas operações de escrita uma única vez se o AWS Glue encontrar um erro de rede.

Conexões do OpenSearch Service

O AWS Glue para Spark pode ser usado para ler e escrever em tabelas no OpenSearch Service no AWS Glue 4.0 e versões posteriores. Você pode definir o que deseja ler do OpenSearch Service com uma consulta do OpenSearch. Conecte-se ao OpenSearch Service usando credenciais de autenticação básica HTTP armazenadas no AWS Secrets Manager via conexão do AWS Glue. Esse recurso não é compatível com o OpenSearch Service com tecnologia sem servidor.

Para obter mais informações sobre o serviço Amazon OpenSearch Service, consulte a Documentação do Amazon OpenSearch Service.

Configurar conexões do OpenSearch Service

Para se conectar ao OpenSearch Service via AWS Glue, você precisará criar e armazenar suas credenciais do OpenSearch Service em um segredo do AWS Secrets Manager e, em seguida, associar esse segredo a uma conexão AWS Glue OpenSearch Service.

Pré-requisitos:

- Identifique o endpoint do domínio, *aosEndpoint* e porta, *aosPort* do qual gostaria de ler, ou crie o recurso seguindo as instruções na documentação do Amazon OpenSearch Service. Para obter mais informações sobre a criação de domínios, consulte Criar e gerenciar domínios do Amazon OpenSearch Service na documentação do Amazon OpenSearch Service.

  Um endpoint de domínio do Amazon OpenSearch Service terá o seguinte formato padrão, https://search-*domainName*-*unstructuredIdContent*.*region*.es.amazonaws.com. Para obter mais informações sobre a identificação do endpoint do domínio, consulte Criar e gerenciar domínios do Amazon OpenSearch Service na documentação do Amazon OpenSearch Service.

  Identifique ou gere credenciais de autenticação básica HTTP, *aosUser* e *aosPassword* para seu domínio.

Para configurar uma conexão com o OpenSearch Service:

1. No AWS Secrets Manager, crie um segredo usando suas credenciais do OpenSearch Service. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *SecretName*, para a próxima etapa.

   - Ao selecionar Pares chave/valor, crie um par para a chave `opensearch.net.http.auth.user` com o valor *aosUser*.
   - Ao selecionar Pares chave/valor, crie um par para a chave `opensearch.net.http.auth.pass` com o valor *aosPassword*.
2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *connectionName*, para uso futuro no AWS Glue.

   - Ao selecionar um Tipo de conexão, escolha OpenSearch Service.
   - Ao selecionar um endpoint de domínio, forneça *aosEndpoint*.

- Ao selecionar uma porta, forneça *aosPort*.
- Ao selecionar um Segredo da AWS, forneça o *secretName*.

Depois de criar uma conexão AWS Glue OpenSearch Service, siga estas etapas antes de executar seu trabalho do AWS Glue:

- Conceda ao perfil do IAM associada ao seu trabalho do AWS Glue permissão para ler *secretName*.
- Na configuração do trabalho do AWS Glue, forneça *connectionName* como uma conexão de rede adicional.

Ler dos índices do OpenSearch Service

Pré-requisitos:

- Um índice do OpenSearch Service do qual você gostaria de ler, *aosIndex*.
- Uma conexão do AWS Glue OpenSearch Service configurada para fornecer informações de autenticação e localização da rede. Para fazer isso, conclua as etapas do procedimento anterior, Para configurar uma conexão com o OpenSearch Service. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Este exemplo lê um índice do Amazon OpenSearch Service. Será necessário fornecer o parâmetro `pushdown`.

Por exemplo:

```
opensearch_read = glueContext.create_dynamic_frame.from_options(
    connection_type="opensearch",
    connection_options={
        "connectionName": "connectionName",
        "opensearch.resource": "aosIndex",
        "pushdown": "true",
    }
)
```

Você também pode fornecer uma srting de consulta para filtrar os resultados retornados em seu DynamicFrame. Você precisará configurar `opensearch.query`.

O `opensearch.query` pode usar uma string de parâmetros de consulta de URL *`queryString`* ou um objeto *`queryObject`* de consulta JSON DSL. Para obter mais informações sobre a consulta DSL, consulte Consulta DSL na documentação do OpenSearch. Para fornecer uma string de parâmetros de consulta de URL, adicione o prefixo `?q=` à sua consulta, como você faria em um URL totalmente qualificado. Para fornecer um objeto DSL de consulta, faça o escape de string do objeto JSON antes de fornecê-lo.

Por exemplo:

```
            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 obter mais informações sobre como criar uma consulta fora de sua sintaxe específica, consulte Sintaxe da string de consulta na documentação do OpenSearch.

Ao ler as coleções do OpenSearch que contêm dados do tipo de matriz, você deve especificar quais campos são do tipo de matriz em sua chamada de método usando o parâmetro `opensearch.read.field.as.array.include`.

Por exemplo, ao ler o documento abaixo, você encontrará os campos `genre` e `actor` de matriz:

```
{
    "_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"
    }
}
```

Nesse caso, você incluiria esses nomes de campo na chamada do método. Por exemplo:

```
"opensearch.read.field.as.array.include": "genre,actor"
```

Se o campo de matriz estiver aninhado dentro da estrutura do documento, consulte-o usando a notação de pontos: `"genre,actor,foo.bar.baz"`. Isso especificaria uma matriz `baz` incluída em seu documento de origem por meio do documento incorporado `foo` que contém o documento incorporado `bar`.

Escrever em tabelas do OpenSearch Service

Este exemplo escreve informações de um DynamicFrame existente, *dynamicFrame*, no OpenSearch Service. Se o índice já tiver informações, o AWS Glue anexará dados do seu DynamicFrame. Será necessário fornecer o parâmetro `pushdown`.

Pré-requisitos:

- Uma tabela do OpenSearch Service na qual você gostaria de escrever. Você precisará de informações de identificação para a tabela. Vamos chamar isso de *tableName*.
- Uma conexão do AWS Glue OpenSearch Service configurada para fornecer informações de autenticação e localização da rede. Para fazer isso, conclua as etapas do procedimento anterior, Para configurar uma conexão com o OpenSearch Service. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Por exemplo:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="opensearch",
    connection_options={
      "connectionName": "connectionName",
      "opensearch.resource": "aosIndex",
    },
)
```

Referência de opções de conexão do OpenSearch Service

- `connectionName` — Obrigatório. Usado para leitura/gravação. O nome de uma conexão do AWS Glue OpenSearch Service configurada para fornecer informações de autenticação e localização da rede ao seu método de conexão.
- `opensearch.resource` — Obrigatório. Usado para leitura/gravação. Valores válidos: nomes de índice do OpenSearch. O nome do índice com o qual seu método de conexão interagirá.
- `opensearch.query` - Usado para leitura. Valores válidos: JSON com escape de string ou, quando essa string começa com ?, a parte de pesquisa de um URL. Uma consulta do OpenSearch que filtra o que deve ser recuperado durante a leitura. Para obter mais informações sobre como usar esse parâmetro, consulte a seção the section called “Ler do OpenSearch Service” anterior.

- `pushdown`: exigido se. Usado para leitura. Valores válidos: booleano. Instrui o Spark a passar consultas de leitura para o OpenSearch para que o banco de dados retorne apenas documentos relevantes.
- `opensearch.read.field.as.array.include`: obrigatório ao ler dados do tipo matriz. Usado para leitura. Valores válidos: listas separadas por vírgulas de nomes de campo. Especifica os campos a serem lidos como matrizes de documentos do OpenSearch. Para obter mais informações sobre como usar esse parâmetro, consulte a seção the section called “Ler do OpenSearch Service” anterior.

Conexões do Redshift

Você pode usar o AWS Glue for Spark para ler e gravar tabelas nos bancos de dados do Amazon Redshift. Ao se conectar aos bancos de dados do Amazon Redshift, o AWS Glue move dados pelo Amazon S3 para obter a máxima taxa de transferência, usando o SQL e os comandos do Amazon Redshift. `COPY UNLOAD` No AWS Glue 4.0 e versões posteriores, você pode usar a integração do Amazon Redshift com o Apache Spark para ler e escrever com otimizações e recursos específicos do Amazon Redshift, além dos disponíveis na conexão por meio de versões anteriores.

Saiba como o AWS Glue está tornando mais fácil do que nunca para os usuários do Amazon Redshift migrarem para o AWS Glue para integração de dados sem servidor e ETL.

Configurar conexões do Redshift

Para usar clusters do Amazon Redshift no AWS Glue, você precisará de alguns pré-requisitos:

- Um diretório do Amazon S3 a ser usado para armazenamento temporário ao ler e gravar no banco de dados.
- Uma Amazon VPC que permite a comunicação entre seu cluster do Amazon Redshift, sua tarefa do AWS Glue e seu diretório do Amazon S3.
- Permissões apropriadas do IAM na tarefa AWS Glue e no cluster Amazon Redshift.

Configurar perfis do IAM

Configurar o perfil para o cluster do Amazon Redshift

Seu cluster do Amazon Redshift precisa ser capaz de ler e gravar no Amazon S3 para se integrar às tarefas do Glue. AWS Para permitir isso, você pode associar perfis do IAM ao cluster do Amazon Redshift ao qual você deseja se conectar. Seu perfil deve ter uma política que permita ler e gravar

no diretório temporário do Amazon S3. Seu perfil deve ter uma relação de confiança que permita ao serviço `redshift.amazonaws.com` `AssumeRole`.

Para associar um perfil do IAM ao Amazon Redshift

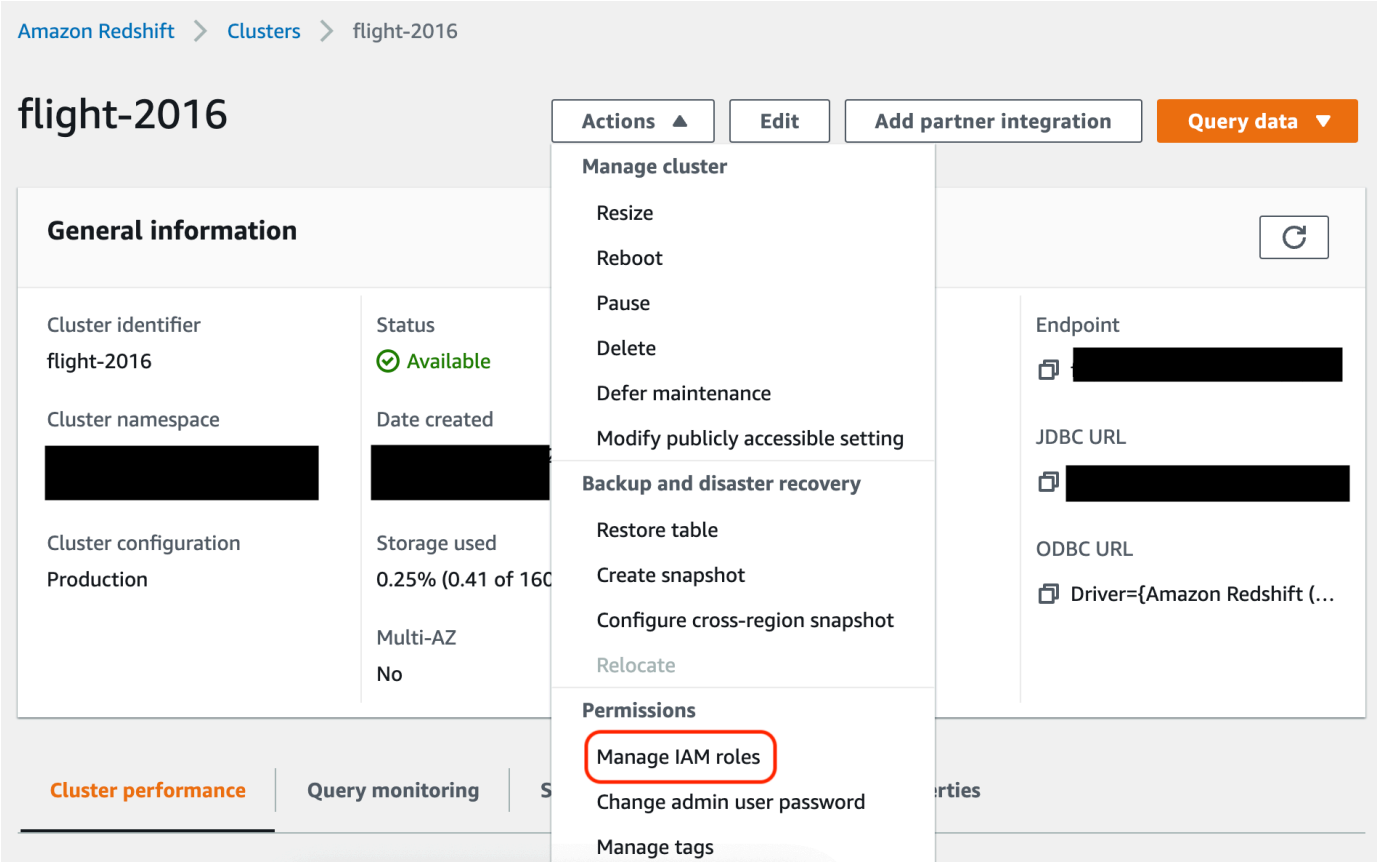
1. Pré-requisitos: um bucket ou diretório do Amazon S3 usado para o armazenamento temporário de arquivos.
2. Identifique quais permissões do Amazon S3 o cluster do Amazon Redshift precisará ter. Ao mover dados de e para um cluster do Amazon Redshift, os trabalhos do AWS Glue emitem instruções COPY e UNLOAD contra o Amazon Redshift. Se seu trabalho modificar uma tabela no Amazon Redshift, o AWS Glue também emitirá instruções CREATE LIBRARY. Para obter informações sobre as permissões específicas do Amazon S3 necessárias para o Amazon Redshift executar essas declarações, consulte a documentação do Amazon Redshift: Amazon Redshift: Permissões para acessar outros recursos. AWS
3. No console do IAM, crie uma política do IAM com as permissões necessárias. Para obter informações sobre a criação de uma política, consulte Criação de políticas do IAM.
4. No console do IAM, crie um perfil e uma relação de confiança que permita ao Amazon Redshift assumir o perfil. Siga as instruções na documentação do IAM Para criar uma função para um AWS serviço (console)

   - Quando solicitado a escolher um caso AWS de uso do serviço, escolha “Redshift - Personalizável”.
   - Quando solicitado a anexar uma política, escolha a política que você definiu anteriormente.

   > **Note**
   >
   > Para obter mais informações sobre a configuração de funções para o Amazon Redshift, consulte Autorizar o Amazon Redshift a AWS acessar outros serviços em seu nome na documentação do Amazon Redshift.

5. No console do Amazon Redshift, associe o perfil ao seu cluster do Amazon Redshift. Siga as instruções na documentação do Amazon Redshift.

   Selecione a opção realçada no console do Amazon Redshift para definir esta configuração:



> **Note**
>
> Por padrão, os trabalhos do AWS Glue passam pelas credenciais temporárias do Amazon Redshift que são criadas usando a função que você especificou para executar o trabalho. Não recomendamos usar esses parâmetros para especificar credenciais. Por motivos de segurança, essas credenciais expiram após 1 hora.

Configurar a função para o trabalho AWS Glue

O trabalho AWS Glue precisa de uma função para acessar o bucket do Amazon S3. Você não precisa de permissões do IAM para o cluster do Amazon Redshift, seu acesso é controlado pela conectividade no Amazon VPC e por suas credenciais do banco de dados.

Configurar a Amazon VCP

Para configurar o acesso aos armazenamentos de dados do Amazon Redshift

1. Faça login no AWS Management Console e abra o console do Amazon Redshift em https://console.aws.amazon.com/redshiftv2/.
2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome de cluster que você deseja acessar no AWS Glue.
4. Na seção Cluster Properties (Propriedades do cluster), escolha um grupo de segurança em VPC security groups (Grupos de segurança da VPC) para permitir que o AWS Glue o utilize. Anote o nome do security group que você escolheu para referência futura. Escolher o grupo de segurança abre a lista Security Groups (Grupos de segurança) do console do Amazon EC2.
5. Escolha o security group a ser modificado e navegue até a guia Inbound.
6. Adicione uma regra de autorreferência para permitir que os componentes do AWS Glue se comuniquem. Especificamente, adicione ou confirme que existe uma regra de Type `All TCP`, que Protocol é `TCP`, Port Range inclui todas as portas e Source e Group ID apresentam o mesmo nome de security group.

   O regra de entrada tem aparência semelhante a esta:

Tipo	Protocolo	Intervalo de portas	Origem
Todos os TCP	TCP	0–65535	database-security-group

   Por exemplo: .

7. Adicione também uma regra para o tráfego de saída. Abra o tráfego de saída a todas as portas, por exemplo:

Tipo	Protocolo	Intervalo de portas	Destino
Todo o tráfego	ALL	ALL	0.0.0.0/0

Ou crie uma regra de autorreferência em que Type (Tipo) é `All TCP`, Protocol (Protocolo) é `TCP`, Port Range (Intervalo de portas) inclui todas as portas e cujo Destination (Destino) é o mesmo nome de security group de Group ID (ID de grupo). Se você estiver usando um endpoint da VPC do Amazon S3, adicione também uma regra HTTPS para acesso ao Amazon S3. O *`s3-prefix-list-id`* é exigido na regra do grupo de segurança para permitir o tráfego da VPC para o endpoint Amazon S3 VPC.

Por exemplo: .

Tipo	Protocolo	Intervalo de portas	Destino
Todos os TCP	TCP	0–65535	*`security-group`*
HTTPS	TCP	443	*`s3- prefix-list-id`*

Configurar o AWS Glue

Você precisará criar uma conexão do AWS Glue Data Catalog que forneça informações de conexão com a Amazon VPC.

Para configurar a conectividade do Amazon Redshift (Amazon VPC) com o AWS Glue no console

1. Crie uma conexão do catálogo de dados seguindo as etapas em:the section called “Adicionar uma conexão do AWS Glue”. Depois de criar a conexão, guarde o nome da conexão, *`connectionName`*, para a próxima etapa.

   - Ao selecionar um tipo de conexão, selecione Amazon Redshift.
   - Ao selecionar um cluster do Redshift, selecione seu cluster pelo nome.
   - Forneça informações de conexão padrão para um usuário do Amazon Redshift em seu cluster.
   - As configurações do Amazon VPC serão definidas automaticamente.

> **Note**
>
> Você precisará fornecer `PhysicalConnectionRequirements` manualmente para a Amazon VPC ao criar uma conexão do Amazon Redshift por meio do SDK da AWS .

2. Na configuração da tarefa do AWS Glue, forneça *`connectionName`* como uma conexão de rede adicional.

Exemplo: leitura de tabelas do Amazon Redshift

Você pode ler nos clusters do Amazon Redshift e ambientes do Amazon Redshift sem servidor.

Pré-requisitos: uma tabela do Amazon Redshift que você deseja ler. Siga as etapas da seção anterior e, em the section called “Configurar o Redshift” seguida, você deverá ter o URI do Amazon S3 para um diretório temporário, *`temp-s3-dir`* e uma função do IAM,, *`rs-role-name`*(na conta). *`role-account-id`*

Using the Data Catalog

Pré-requisitos adicionais: um banco de dados e uma tabela do catálogo de dados para a tabela do Amazon Redshift que você deseja ler. Para obter mais informações sobre o catálogo de dados, consulte Descoberta e catalogação de dados. Depois de criar uma entrada para sua tabela do Amazon Redshift, você identificará sua conexão com um e. *`redshift-dc-database-nameredshift-table-name`*

Configuração: nas opções de função, você identificará a tabela do catálogo de dados com os parâmetros `database` e `table_name`. Você identificará seu diretório temporário do Amazon S3 com `redshift_tmp_dir`. Você também fornecerá *`rs-role-name`*usando a `aws_iam_role` chave no `additional_options` parâmetro.

```
 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

Pré-requisitos adicionais: Você precisará do nome da sua tabela do Amazon Redshift (. *redshift-table-name* Você precisará das informações da conexão JDBC para o cluster do Amazon Redshift que armazena essa tabela. Você fornecerá suas informações de conexão com *host*, *porta redshift-database-name*, *nome de usuário* e *senha*.

Você pode recuperar suas informações de conexão no console do Amazon Redshift ao trabalhar com clusters do Amazon Redshift. Ao usar o Amazon Redshift sem servidor, consulte Conectar-se ao Amazon Redshift Serverless na documentação do Amazon Redshift.

Configuração: nas opções de função, você identificará os parâmetros `url`, `dbtable`, `user` e `password`. Você identificará seu diretório temporário do Amazon S3 com `redshift_tmp_dir`. Você pode especificar seu perfil do IAM usando `aws_iam_role` quando você usa `from_options`. A sintaxe é semelhante à conexão por meio do catálogo de dados, mas você coloca os parâmetros no mapa de `connection_options`.

É uma má prática codificar senhas em scripts AWS Glue. Considere armazenar suas senhas AWS Secrets Manager e recuperá-las em seu script com o 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)
```

Exemplo: gravaçr em tabelas do Amazon Redshift

Você pode ler dos clusters do Amazon Redshift e em ambientes do Amazon Redshift sem servidor.

Pré-requisitos: Um cluster do Amazon Redshift e siga as etapas na seção anterior. the section called “Configurar o Redshift” Depois disso, você deve ter o URI do Amazon S3 para um diretório temporário, *temp-s3-dir* e uma função do IAM,, (na conta). *rs-role-namerole-account-*

id Você também precisará de um `DynamicFrame` cujo conteúdo você deseja gravar no banco de dados.

Using the Data Catalog

Pré-requisitos adicionais: um banco de dados do catálogo de dados para o cluster do Amazon Redshift no qual que você deseja gravar. Para obter mais informações sobre o catálogo de dados, consulte Descoberta e catalogação de dados. Você identificará sua conexão *redshift-dc-database-name*e a tabela de destino com *redshift-table-name*.

Configuração: nas opções de função, você identificará o banco de dados do catálogo de dados com o parâmetros `database` e depois fornecerá a tabela com `table_name`. Você identificará seu diretório temporário do Amazon S3 com `redshift_tmp_dir`. Você também fornecerá *rs-role-name*usando a `aws_iam_role` chave no `additional_options` parâmetro.

```
 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

Você pode se conectar ao Amazon Redshift diretamente usando o método `write_dynamic_frame.from_options`. Porém, em vez de inserir os detalhes da conexão diretamente no script, você pode referenciar os detalhes da conexão armazenados em uma conexão do catálogo de dados com o método `from_jdbc_conf`. Você pode fazer isso sem fazer crawling ou criar tabelas do catálogo de dados para o banco de dados. Para obter mais informações sobre as conexões do catálogo de dados, consulte Conectar a dados.

Pré-requisitos adicionais: uma conexão do catálogo de dados para seu banco de dados, uma tabela do Amazon Redshift que você deseja ler

Configuração: você identificará sua conexão com o Data Catalog *dc-connection-name*. Você identificará seu banco de dados e tabela do Amazon Redshift com e. *redshift-table-nameredshift-database-name* Você fornecerá as informações de conexão do catálogo de dados com `catalog_connection` e as informações do Amazon Redshift com `dbtable` e

`database`. A sintaxe é semelhante à conexão por meio do catálogo de dados, mas você coloca os parâmetros no mapa de `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"])
```

Referência da opções de conexão do Amazon Redshift

As opções básicas de conexão usadas em todas as conexões JDBC do AWS Glue para configurar informações com`ourl`, `user` e `password` são consistentes em todos os tipos de JDBC. Para obter mais informações sobre os parâmetros padrão de JDBC consulte the section called “Parâmetros de conexão JDBC”.

O tipo de conexão do Amazon Redshift tem algumas opções adicionais de conexão:

- `"redshiftTmpDir"`: (obrigatório) o caminho do Amazon S3 onde dados temporários podem ser preparados ao serem copiados para fora do banco de dados.
- `"aws_iam_role"`: (opcional) o ARN de um perfil do IAM. O trabalho AWS Glue passará essa função para o cluster do Amazon Redshift para conceder ao cluster as permissões necessárias para concluir as instruções do trabalho.

Opções adicionais de conexão disponíveis no AWS Glue 4.0+

Você também pode transmitir opções para o novo conector Amazon Redshift por meio das opções de conexão AWS Glue. Para obter uma lista completa das opções de conectores compatíveis, consulte a seção de Parâmetros do Spark SQL em Amazon Redshift integration for Apache Spark (Integração do Amazon Redshift for Apache Spark).

Para sua conveniência, reiteramos algumas novas opções aqui:

Nome	Obrigatório	Padrão	Descrição
autopushdown	Não	VERDADEIRO	Aplica pushdown de predicados e consultas capturando e analisando os planos lógicos do Spark para operações SQL. As operações são traduzidas em uma consulta SQL e executadas no Amazon Redshift para melhorar a performance.
autopushdown.s3_result_cache	Não	FALSE	Armazena a consulta SQL para descarregar dados do mapeamento de caminhos do Amazon S3 na memória para que a mesma consulta não precise ser executada novamente na mesma sessão do Spark. Só é possível quando o `autopushdown` está habilitado.
unload_s3_format	Não	PARQUET	PARQUET - descarrega os resultados da consulta no formato Parquet.

Nome	Obrigatório	Padrão	Descrição
			TEXT - descarreg a os resultados da consulta em formato de texto delimitado por barras.
sse_kms_key	Não	N/D	A chave AWS SSE-KMS a ser usada para criptografia durante UNL0AD as operações , em vez da criptogra fia padrão para. AWS

Nome	Obrigatório	Padrão	Descrição
extracopyoptions	Não	N/D	Uma lista de opções extras para anexar ao comando `COPY` do Amazon Redshift ao carregar dados, como `TRUNCATECOLUMNS` ou `MAXERROR n` (para outras opções, consulte COPY: parâmetros opcionais). Observe que, como essas opções são anexadas ao fim do comando `COPY`, apenas opções que façam sentido no final do comando podem ser usadas. Isso deve abranger a maioria dos casos de uso possíveis.
csvnullstring (experimental)	Não	NULL	O valor da string a ser gravado para nulos ao usar o `tempformat` CSV. Esse deve ser um valor que não apareça nos dados reais.

Esses novos parâmetros podem ser usados como se segue.

Novas opções para melhoria de performance

O novo conector introduz algumas novas opções de melhoria de performance:

- `autopushdown`: habilitado por padrão.
- `autopushdown.s3_result_cache`: desabilitado por padrão.
- `unload_s3_format`: `PARQUET` por padrão.

Para obter informações sobre como usar essas opções, consulte Amazon Redshift integration for Apache Spark (Integração do Amazon Redshift for Apache Spark). Recomendamos que você não ative o `autopushdown.s3_result_cache` quando tiver operações mistas de leitura e gravação, pois os resultados em cache podem conter informações obsoletas. A opção `unload_s3_format` é definida como `PARQUET` por padrão para o comando `UNLOAD`, para melhorar a performance e reduzir o custo de armazenamento. Para usar o comportamento padrão do comando `UNLOAD`, redefina a opção como `TEXT`.

Nova opção de criptografia para leitura

Por padrão, os dados na pasta temporária usada pelo AWS Glue ao ler dados da tabela do Amazon Redshift são criptografados usando a criptografia `SSE-S3`. Para usar as chaves gerenciadas pelo cliente de AWS Key Management Service (AWS KMS) para criptografar seus dados, você pode configurar de `("sse_kms_key" # kmsKey)` onde KSMKey é o ID da chave AWS KMS, em vez da opção de configuração herdada `("extraunloadoptions" # s"ENCRYPTED KMS_KEY_ID '$kmsKey'")` na AWS Glue versão 3.0.

```
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"
)
```

Compatibilidade com a URL JDBC baseada no IAM

O novo conector é compatível com uma URL JDBC baseada no IAM para que você não precise passar um usuário/senha ou um segredo. Com uma URL JDBC baseada no IAM, o conector usa o perfil de runtime de trabalho para acessar a fonte de dados do Amazon Redshift.

Etapa 1: anexar a seguinte política mínima exigida ao perfil de runtime 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": "*"
        }
    ]
}
```

Etapa 2: usar a URL do JDBC baseado em IAM como se segue. Especifique uma nova opção `DbUser` com o nome de usuário do Amazon Redshift com o qual você está se 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**
>
> Atualmente, um `DynamicFrame` é compatível apenas com uma URL JDBC baseada no IAM com um `DbUser` no fluxo de trabalho `GlueContext.create_dynamic_frame.from_options`.

Migrar do AWS Glue versão 3.0 para a versão 4.0

No AWS Glue 4.0, as tarefas de ETL têm acesso a um novo conector Amazon Redshift Spark e a um novo driver JDBC com opções e configurações diferentes. O novo conector e o novo driver do Amazon Redshift foram criados pensando em performance e mantêm a consistência transacional dos dados. Esses produtos estão documentados na documentação do Amazon Redshift. Para obter mais informações, consulte:

- Integração do Amazon Redshift for Apache Spark
- Driver JDBC do Amazon Redshift, versão 2.1

Restrição de nomes e identificadores de tabelas/colunas

O novo conector e driver do Amazon Redshift Spark têm um requisito mais restrito para o nome da tabela do Redshift. Para obter mais informações, consulte Nomes e identificadores para definir o nome da tabela do Amazon Redshift. O fluxo de trabalho de marcadores de trabalho pode não funcionar com um nome de tabela que não corresponda às regras e com determinados caracteres, como espaço.

Se você tiver tabelas antigas com nomes que não estão em conformidade com as regras de nomes e identificadores e tiver problemas com marcadores (trabalhos que reprocessam dados de tabelas antigas do Amazon Redshift), recomendamos que você renomeie as tabelas. Para obter mais informações, consulte Exemplos de ALTER TABLE.

Alteração de tempformat padrão no Dataframe

O conector Spark versão 3.0 da AWS Glue usa como padrão o `tempformat` para CSV ao gravar no Amazon Redshift. Para ser consistente, no AWS Glue versão 3.0, o `DynamicFrame` ainda tem como padrão que o `tempformat` use `CSV`. Se você já usou APIs de Dataframe do Spark

diretamente com o conector do Amazon Redshift Spark, você pode definir explicitamente o `tempformat` como CSV nas opções `DataframeReader/Writer`. Caso contrário, o padrão para `tempformat` é `AVRO` no novo conector Spark.

Alteração de comportamento: mapeie o tipo de dados REAL do Amazon Redshift para o tipo de dados do FLOAT do Spark, em vez de DOUBLE

No AWS Glue versão 3.0, o Amazon Redshift `REAL` é convertido em um tipo `DOUBLE` do Spark. O novo conector do Amazon Redshift Spark atualizou o comportamento para que o tipo `REAL` do Amazon Redshift seja convertido para o tipo `FLOAT` do Spark e vice-versa. Se você tiver um caso de uso antigo em que ainda desejar que o tipo `REAL` do Amazon Redshift seja mapeado para um tipo `DOUBLE` do Spark, poderá usar a seguinte solução alternativa:

- Para um `DynamicFrame`, mapeie o tipo `Float` para um tipo `Double` com `DynamicFrame.ApplyMapping`. Para um `Dataframe`, você precisa usar `cast`.

Exemplo de código:

```
dyf_cast = dyf.apply_mapping([('a', 'long', 'a', 'long'), ('b', 'float', 'b',
 'double')])
```

Conexões do Kafka

Designa uma conexão com um cluster do Kafka ou um cluster do Amazon Managed Streaming for Apache Kafka.

É possível ler e gravar fluxos de dados do Kafka usando informações armazenadas em uma tabela do Catálogo de Dados ou fornecendo informações para acessar diretamente o fluxo de dados. Você pode ler informações de Kafka em um Spark e depois DataFrame convertê-las em um Glue. AWS DynamicFrame Você pode DynamicFrames escrever no Kafka no formato JSON. Se você acessar diretamente o fluxo de dados, use essas opções para fornecer as informações sobre como acessar o fluxo de dados.

Se você usar `getCatalogSource` ou `create_data_frame_from_catalog` para consumir registros de uma fonte de streaming do Kafka, ou `getCatalogSink` ou `write_dynamic_frame_from_catalog` para gravar registros no Kafka, e o trabalho tiver o banco de dados do catálogo de dados e as informações de nome da tabela, e poderá usá-los para obter alguns parâmetros básicos para leitura da fonte de streaming do Kafka. Se você usar `getSource`, `getCatalogSink`, `getSourceWithFormat`,

`getSinkWithFormat`, `createDataFrameFromOptions`, `create_data_frame_from_options` ou `write_dynamic_frame_from_catalog`, será necessário especificar esses parâmetros básicos usando as opções de conexão descritas aqui.

Você pode especificar as opções de conexão para o Kafka usando os seguintes argumentos para os métodos especificados na classe `GlueContext`.

- Scala
  - `connectionOptions`: usar com `getSource`, `createDataFrameFromOptions`, `getSink`
  - `additionalOptions`: usar com `getCatalogSource`, `getCatalogSink`
  - `options`: usar com `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: usar com `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: usar com `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: usar com `getSource`, `getSink`

Para notas e restrições sobre trabalhos de ETL de streaming, consulte the section called “Notas e restrições sobre ETL de transmissão”.

Configurar o Kafka

Não há AWS pré-requisitos para se conectar aos streams do Kafka disponíveis pela Internet.

Você pode criar uma conexão AWS Glue Kafka para gerenciar suas credenciais de conexão. Para ter mais informações, consulte the section called “Criar uma conexão para um fluxo de dados do Kafka”. Na configuração do trabalho do AWS Glue, forneça *`connectionName`* como uma conexão de rede adicional e, em seguida, na chamada do método, forneça *`connectionName`* ao parâmetro. `connectionName`

Em certos casos, você precisará configurar pré-requisitos adicionais:

- Se estiver usando o Amazon Managed Streaming for Apache Kafka com autenticação do IAM, você precisará da configuração apropriada do IAM.
- Se estiver usando o Amazon Managed Streaming for Apache Kafka com uma Amazon VPC, você precisará da configuração apropriada do Amazon VPC. Você precisará criar uma conexão

AWS Glue que forneça informações de conexão com a Amazon VPC. Você precisará que sua configuração de trabalho inclua a conexão AWS Glue como uma conexão de rede adicional.

Para obter mais informações sobre pré-requisitos de trabalho de ETL de streaming, consulte the section called “Trabalhos de transmissão de ETL”.

Exemplo: leitura de fluxos do Kafka

Usado em conjunto com the section called “forEachBatch”.

Exemplo para fonte de transmissão do 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)
```

Exemplo: gravação em fluxos do Kafka

Exemplos de gravação no Kafka:

Exemplo com o 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()
```

Exemplo com o 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)
```

Referência de opções de conexão do Kafka

Ao ler, use as seguintes opções de conexão com `"connectionType": "kafka"`:

- `"bootstrap.servers"` (obrigatório) uma lista de URLs do servidor de bootstrap, por exemplo, como `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`. Essa opção deve ser especificada na chamada de API ou definida nos metadados da tabela no Data Catalog.
- `"security.protocol"` (obrigatório) O protocolo usado para se comunicar com os agentes. Os valores possíveis são `"SSL"` ou `"PLAINTEXT"`.
- `"topicName"` (Obrigatório): uma lista separada por vírgulas de tópicos para assinar. É necessário especificar um e apenas um de `"topicName"`, `"assign"` ou `"subscribePattern"`.
- `"assign"` (Obrigatório): uma string JSON para especificar o `TopicPartitions` a ser consumido. É necessário especificar um e apenas um de `"topicName"`, `"assign"` ou `"subscribePattern"`.

  Exemplo: '{"topicA":[0,1],"topicB":[2,4]}'
- `"subscribePattern"`: (obrigatório) uma string regex Java que identifica a lista de tópicos para assinar. É necessário especificar um e apenas um de `"topicName"`, `"assign"` ou `"subscribePattern"`.

  Exemplo: 'topic.*'
- `"classification"` (Obrigatório) O formato de arquivo usado pelos dados no registro. Obrigatório, a menos que fornecido por meio do catálogo de dados.
- `"delimiter"` (Opcional) O separador de valores usado quando a `classification` é CSV. O padrão é `","`.
- `"startingOffsets"`: (opcional) a posição inicial no tópico do Kafka de onde ler os dados. Os valores possíveis são `"earliest"` ou `"latest"`. O valor padrão é `"latest"`.

- `"startingTimestamp"`: (Opcional, compatível somente com o AWS Glue versão 4.0 ou posterior) O timestamp do registro no tópico do Kafka do qual ler os dados. O valor possível é uma string de timestamp no formato UTC no padrão `yyyy-mm-ddTHH:MM:SSZ` (onde `Z` representa um desvio do fuso horário UTC com +/-). Por exemplo: "2023-04-04T08:00:00-04:00").

  Observação: somente um dos 'startingOffsets' ou 'startingTimestamp' pode estar presente na lista de opções de conexão do script de streaming AWS Glue. Incluir essas duas propriedades resultará em falha no trabalho.
- `"endingOffsets"`: (opcional) o ponto final quando uma consulta em lote é encerrada. Os valores possíveis são `"latest"` ou uma string JSON que especifica um deslocamento final para cada `TopicPartition`.

  Para a string JSON, o formato é `{"topicA":{"0":23,"1":-1},"topicB":{"0":-1}}`. O valor `-1` como um deslocamento representa `"latest"`.
- `"pollTimeoutMs"`: (opcional) o tempo limite em milissegundos para sondar dados do Kafka em executores de trabalho do Spark. O valor padrão é `512`.
- `"numRetries"`: (opcional) o número de novas tentativas antes de falhar em obter os deslocamentos do Kafka. O valor padrão é `3`.
- `"retryIntervalMs"`: (opcional) o tempo em milissegundos a se esperar antes de tentar novamente buscar os deslocamentos do Kafka. O valor padrão é `10`.
- `"maxOffsetsPerTrigger"`: (opcional) o limite de taxa no número máximo de deslocamentos que são processados por intervalo do acionador. O número total especificado de deslocamentos é dividido proporcionalmente entre `topicPartitions` de diferentes volumes. O valor padrão é nulo, o que significa que o consumidor lê todos os deslocamentos até o deslocamento mais recente conhecido.
- `"minPartitions"`: (opcional) o número mínimo desejado de partições a serem lidas do Kafka. O valor padrão é nulo, o que significa que o número de partições do Spark é igual ao número de partições do Kafka.
- `"includeHeaders"`: (opcional) informa se deve incluir os cabeçalhos do Kafka. Quando a opção estiver definida como "true", a saída de dados conterá uma coluna adicional chamada "glue_streaming_kafka_headers" com o tipo `Array[Struct(key: String, value: String)]`. O valor padrão é “false”. Essa opção está disponível no AWS Glue versão 3.0 ou posterior.
- `"schema"` (Obrigatório quando inferSchema é definido como false): o esquema a ser usado para processar a carga útil. Se a classificação for `avro`, o esquema fornecido deverá estar no formato

de esquema Avro. Se a classificação não for `avro`, o esquema fornecido deverá estar no formato de esquema DDL.

Veja a seguir alguns exemplos de esquema.

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): o valor padrão é "false". Se definido como “true”, o esquema será detectado em runtime com base na carga útil em `foreachbatch`.
- `"avroSchema"` (Descontinuado): parâmetro usado para especificar um esquema de dados Avro quando o formato Avro é usado. Esse parâmetro foi descontinuado. Use o parâmetro `schema`.
- `"addRecordTimestamp"`: (opcional) quando essa opção for definida como “true“, a saída de dados conterá uma coluna adicional denominada “__src_timestamp” que indica a hora em que o

registro correspondente foi recebido pelo tópico. O valor padrão é "false". Essa opção é compatível com o AWS Glue versão 4.0 ou posterior.

- `"emitConsumerLagMetrics"`: (Opcional) Quando a opção for definida como 'verdadeira', para cada lote, ela emitirá as métricas da duração entre o registro mais antigo recebido pelo tópico e o horário em AWS Glue que ele chegou. CloudWatch O nome da métrica é “glue.driver.streaming”. maxConsumerLagInMs”. O valor padrão é "false". Essa opção é compatível com o AWS Glue versão 4.0 ou posterior.

Ao gravar, use as seguintes opções de conexão com o `"connectionType": "kafka"`:

- `"connectionName"`(Obrigatório) Nome da conexão AWS Glue usada para se conectar ao cluster Kafka (semelhante à fonte do Kafka).
- `"topic"` (Obrigatório) Se uma coluna de tópico existir, seu valor será usado como tópico ao gravar a linha especificada no Kafka, a menos que a opção de configuração do tópico esteja definida. Ou seja, a opção de configuração `topic` substitui a coluna do tópico.
- `"partition"` (Opcional) Se um número de partição válido for especificado, essa `partition` será usada no envio do registro.

  Se nenhuma partição for especificada, mas uma `key` estiver presente, uma partição será escolhida usando um hash da chave.

  Se nem `key` nem `partition` estiverem presentes, uma partição será escolhida com base no particionamento fixo dessas alterações quando pelo menos bytes batch.size forem produzidos na partição.
- `"key"` (Opcional) Usado para particionar se `partition` for nulo.
- `"classification"` (Opcional) O formato de arquivo usado pelos dados no registro. Só oferecemos suporte a JSON, CSV e Avro.

  Com o formato Avro, podemos fornecer um avroSchema personalizado para serializar, mas observe que isso também precisa ser fornecido na fonte para desserialização. Caso contrário, por padrão, ele usa o Apache AvroSchema para serializar.

Além disso, você pode ajustar o coletor Kafka conforme necessário atualizando os parâmetros de configuração do produtor Kafka. Observe que não há nenhuma lista de permissões nas opções de conexão. Todos os pares de chave-valor são mantidos no coletor como estão.

No entanto, há uma pequena lista de opções negadas que não entrarão em vigor. Para obter mais informações, consulte Configurações específicas do Kafka.

Conexões do Azure Cosmos DB

O AWS Glue para Spark pode ser usado para ler e escrever em contêineres existentes no Azure Cosmos DB usando a API NoSQL no AWS Glue 4.0 e versões posteriores. Você pode definir o que ler no Azure Cosmos DB com uma consulta SQL. Conecte-se ao Azure Cosmos DB usando uma chave do Azure Cosmos DB armazenada no AWS Secrets Manager via conexão do AWS Glue.

Para obter mais informações sobre o Azure Cosmos DB para NoSQL, consulte a Documentação do Azure.

Configurar conexões do Azure Cosmos DB

Para se conectar ao Azure Cosmos DB via AWS Glue, será necessário criar e armazenar sua chave do Azure Cosmos DB em um segredo do AWS Secrets Manager e, em seguida, associar esse segredo a uma conexão do Azure Cosmos DB AWS Glue.

Pré-requisitos:

- No Azure, será necessário identificar ou gerar uma chave do Azure Cosmos DB para uso pelo AWS Glue, `cosmosKey`. Para obter mais informações, consulte Acesso seguro a dados no Azure Cosmos DB na documentação do Azure.

Para configurar uma conexão com o Azure Cosmos DB:

1. No AWS Secrets Manager, crie um segredo usando sua chave do Azure Cosmos DB. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *`SecretName`*, para a próxima etapa.

   - Ao selecionar Pares chave/valor, crie um par para a chave `spark.cosmos.accountKey` com o valor *`cosmosKey`*.
2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *`connectionName`*, para uso futuro no AWS Glue.

   - Ao selecionar um Tipo de conexão, selecione Azure Cosmos DB.
   - Ao selecionar um Segredo da AWS, forneça o *`secretName`*.

Depois de criar uma conexão AWS Glue Azure Cosmos DB, siga estas etapas antes de executar seu trabalho do AWS Glue:

- Conceda ao perfil do IAM associada ao seu trabalho do AWS Glue permissão para ler *secretName*.
- Na configuração do trabalho do AWS Glue, forneça *connectionName* como uma conexão de rede adicional.

Ler de contêineres do Azure Cosmos DB para NoSQL

Pré-requisitos:

- Um contêiner do Azure Cosmos DB para NoSQL do qual você gostaria de ler. Você precisará de informações de identificação para o contêiner.

  Um contêiner do Azure Cosmos para NoSQL é identificado por seu banco de dados e contêiner. É necessário fornecer os nomes do banco de dados, *cosmosDBName* e do contêiner, *cosmosContainerName*, ao se conectar à API do Azure Cosmos para NoSQL.
- Uma conexão do AWS Glue Azure Cosmos DB configurada para fornecer informações de autenticação e localização da rede. Para fazer isso, conclua as etapas do procedimento anterior, Para configurar uma conexão com o Azure Cosmos DB. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Por exemplo:

```
azurecosmos_read = glueContext.create_dynamic_frame.from_options(
    connection_type="azurecosmos",
    connection_options={
    "connectionName": connectionName,
    "spark.cosmos.database": cosmosDBName,
    "spark.cosmos.container": cosmosContainerName,
    }
)
```

Você também pode fornecer uma consulta SQL SELECT para filtrar os resultados retornados ao seu DynamicFrame. Você precisará configurar `query`.

Por exemplo:

```
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"
    }
)
```

Escrever em contêineres do Azure Cosmos DB para NoSQL

Este exemplo escreve informações de um DynamicFrame existente, *dynamicFrame*, no Azure Cosmos DB. Se o contêiner já contiver informações, o AWS Glue anexará dados do seu DynamicFrame. Se as informações no contêiner tiverem um esquema diferente das informações que você escrever, haverá erros.

Pré-requisitos:

- Uma tabela do Azure Cosmos DB em que você deseja escrever. Você precisará de informações de identificação para o contêiner. É necessário criar o contêiner antes de chamar o método de conexão.

  Um contêiner do Azure Cosmos para NoSQL é identificado por seu banco de dados e contêiner. É necessário fornecer os nomes do banco de dados, *cosmosDBName* e do contêiner, *cosmosContainerName*, ao se conectar à API do Azure Cosmos para NoSQL.
- Uma conexão do AWS Glue Azure Cosmos DB configurada para fornecer informações de autenticação e localização da rede. Para fazer isso, conclua as etapas do procedimento anterior, Para configurar uma conexão com o Azure Cosmos DB. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Por exemplo:

```
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
```

```
)
```

Referência de opções de conexão do Azure Cosmos DB

- `connectionName` — Obrigatório. Usado para leitura/gravação. O nome de uma conexão AWS Glue Azure Cosmos DB configurada para fornecer informações de autenticação e localização da rede ao seu método de conexão.
- `spark.cosmos.database` — Obrigatório. Usado para leitura/gravação. Valores válidos: nomes de bancos de dados. Nome do banco de dados do Azure Cosmos DB para NoSQL.
- `spark.cosmos.container` — Obrigatório. Usado para leitura/gravação. Valores válidos: nomes de contêineres. Nome do contêiner do Azure Cosmos DB para NoSQL.
- `spark.cosmos.read.customQuery` - Usado para leitura. Valores válidos: consultas SQL SELECT. Consulta personalizada para selecionar documentos a serem lidos.

Conexões do Azure SQL

O AWS Glue para Spark pode ser usado para ler e gravar em tabelas em instâncias gerenciadas do Azure SQL no AWS Glue 4.0 e versões posteriores. Você pode definir o que ler no Azure SQL com uma consulta SQL. Conecte-se ao Azure SQL usando credenciais de usuário e senha armazenadas no AWS Secrets Manager via conexão do AWS Glue.

Para obter mais informações sobre o Azure SQL, consulte a Documentação do Azure SQL.

Configurar conexões do Azure SQL

Para se conectar ao Azure SQL via AWS Glue, será necessário criar e armazenar sua credenciais do Azure SQL em um segredo do AWS Secrets Manager e, em seguida, associar esse segredo a uma conexão do Azure SQL AWS Glue.

Para configurar uma conexão com o Azure SQL:

1. No AWS Secrets Manager, crie um segredo usando suas credenciais do Azure SQL. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *`SecretName`*, para a próxima etapa.
   - Ao selecionar Pares chave/valor, crie um par para a chave `user` com o valor *`azuresqlUsername`*.

- Ao selecionar Pares chave/valor, crie um par para a chave `password` com o valor *azuresqlPassword*.

2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *connectionName*, para uso futuro no AWS Glue.

   - Ao selecionar um Tipo de conexão, selecione Azure SQL.
   - Ao fornecer o URL do Azure SQL, forneça um URL de endpoint do JDBC.

     Essa lista deve estar no seguinte formato: `jdbc:sqlserver://`*databaseServerName*`:`*databasePort*`;databaseName=`*azuresqlDBnam*

     O AWS Glue requer as seguintes propriedades de URL:

     - `databaseName`: um banco de dados padrão no Azure SQL ao qual se conectar.

     Para obter mais informações sobre URLs de JDBC para instâncias gerenciadas Azure SQL, consulte a Documentação da Microsoft.
   - Ao selecionar um Segredo da AWS, forneça o *secretName*.

Depois de criar uma conexão AWS Glue Azure SQL, siga estas etapas antes de executar seu trabalho do AWS Glue:

- Conceda ao perfil do IAM associada ao seu trabalho do AWS Glue permissão para ler *secretName*.
- Na configuração do trabalho do AWS Glue, forneça *connectionName* como uma conexão de rede adicional.

Ler de tabelas SQL do Azure

Pré-requisitos:

- Uma tabela do Azure SQL da qual você deseja ler. Você precisará de informações de identificação para a tabela, *databaseName* e *tableIdentifier*.

  Uma tabela do Azure SQL é identificada por seu banco de dados, esquema e nome da tabela. É necessário fornecer o nome do banco de dados e o nome da tabela ao se conectar ao Azure SQL. Você também deverá fornecer o esquema se ele não for o padrão, "public". O banco de dados

é fornecido por meio de uma propriedade de URL em *connectionName*, esquema e nome da tabela via `dbtable`.

- Uma conexão AWS Glue Azure SQL configurada para fornecer informações de autenticação. Conclua as etapas do procedimento anterior, Para configurar uma conexão com o Azure SQL para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Por exemplo:

```
azuresql_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier"
    }
)
```

Você também pode fornecer uma consulta SQL SELECT para filtrar os resultados retornados ao seu DynamicFrame. Você precisará configurar `query`.

Por exemplo:

```
azuresql_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

Escrevendo em tabelas do Azure SQL

Este exemplo escreve informações de um DynamicFrame existente, *dynamicFrame*, no Azure SQL. Se a tabela já contiver informações, o AWS Glue anexará dados do seu DynamicFrame.

Pré-requisitos:

- Uma tabela do Azure SQL em que você deseja escrever. Você precisará de informações de identificação para a tabela, *databaseName* e *tableIdentifier*.

Uma tabela do Azure SQL é identificada por seu banco de dados, esquema e nome da tabela. É necessário fornecer o nome do banco de dados e o nome da tabela ao se conectar ao Azure SQL. Você também deverá fornecer o esquema se ele não for o padrão, "public". O banco de dados é fornecido por meio de uma propriedade de URL em *connectionName*, esquema e nome da tabela via `dbtable`.

- Informações de autenticação do Azure SQL. Conclua as etapas do procedimento anterior, Para configurar uma conexão com o Azure SQL para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Por exemplo:

```
azuresql_write = glueContext.write_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier"
    }
)
```

Referência de opções de conexão do Azure SQL

- `connectionName` — Obrigatório. Usado para leitura/gravação. O nome de uma conexão AWS Glue Azure SQL configurada para fornecer informações de autenticação e localização da rede ao seu método de conexão.
- `databaseName`: usado para leitura/gravação. Valores válidos: nomes de bancos de dados do Azure SQL. O nome do banco de dados do Azure SQL ao qual conectar.
- `dbtable`: necessário para escrever, obrigatório para leitura, a menos que `query` seja fornecido. Usado para leitura/gravação. Valores válidos: nomes de tabelas do Azure SQL ou combinações de nomes de esquema/tabela separados por pontos. Usado para especificar a tabela e o esquema que identificam a tabela à qual conectar. O esquema padrão é "public". Se sua tabela estiver em um esquema não padrão, forneça essas informações no formulário *schemaName*`.`*tableName*.
- `query` - Usado para leitura. Uma consulta SQL SELECT que define o que deve ser recuperado na leitura do Azure SQL. Para obter mais informações, consulte a Documentação da Microsoft.

Conexões BigQuery

Você pode usar o AWS Glue for Spark para ler e gravar tabelas no Google BigQuery no AWS Glue 4.0 e versões posteriores. Você pode ler no BigQuery com uma consulta do Google SQL. Você se conecta ao BigQuery usando credenciais armazenadas AWS Secrets Manager por meio de uma conexão AWS Glue.

Para obter mais informações sobre o Google BigQuery, consulte o site do Google Cloud BigQuery.

Configurar conexões BigQuery

Para se conectar ao Google BigQuery a partir do AWS Glue, você precisará criar e armazenar suas credenciais do Google Cloud Platform em um segredo e, em seguida, associar esse AWS Secrets Manager segredo a uma conexão do Google BigQuery AWS Glue.

Para configurar uma conexão com o BigQuery:

1. No Google Cloud Platform, crie e identifique recursos relevantes:
   - Crie ou identifique um projeto do GCP contendo tabelas do BigQuery às quais você gostaria de se conectar.
   - Ative a API BigQuery. Para obter mais informações, consulte Use the BigQuery Storage Read API to read na tabela.
2. No Google Cloud Platform, crie e exporte as credenciais da conta de serviço:

   Você pode usar o assistente de credenciais do BigQuery para acelerar essa etapa: criar credenciais.

   Para criar uma conta de serviço no GCP, siga o tutorial disponível em Criar contas de serviço.
   - Ao selecionar o projeto, selecione o projeto que contém sua tabela do BigQuery.
   - Ao selecionar perfis do IAM do GCP para sua conta de serviço, adicione ou crie um papel que conceda permissões apropriadas para executar jobs do BigQuery para ler, gravar ou criar tabelas do BigQuery.

   Para criar credenciais para a sua conta de serviço, siga o tutorial disponível em Criar uma chave da conta de serviço.
   - Ao selecionar o tipo de chave, selecione JSON.

Agora você deve ter baixado um arquivo JSON com credenciais para sua conta de serviço. A aparência deve ser semelhante à seguinte:

```
{
  "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. base64 codifica seu arquivo de credenciais baixado. Em uma AWS CloudShell sessão ou similar, você pode fazer isso na linha de comando executando `cat` *`credentialsFile.json`* `| base64 -w 0`. Mantenha a saída desse comando, *`credentialString`*.
4. Nos AWS Secrets Manager, crie um segredo usando suas credenciais do Google Cloud Platform. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *`SecretName`*, para a próxima etapa.
   - *Ao selecionar* ***pares de chave/valor****, crie um par para a chave* *credentials com o valor credentialString.*
5. No catálogo de dados do AWS Glue, crie uma conexão seguindo as etapas em the section called “Adicionar uma conexão do AWS Glue”. Depois de criar a conexão, guarde o nome da conexão, *`connectionName`*, para a próxima etapa.
   - Ao selecionar um tipo de conexão, selecione Google BigQuery.
   - Ao selecionar um Segredo da AWS, forneça o *`secretName`*.
6. Conceda ao perfil do IAM associada ao seu trabalho do AWS Glue permissão para ler *`secretName`*.
7. Na configuração do trabalho do AWS Glue, forneça *`connectionName`* como uma conexão de rede adicional.

Ler das tabelas do BigQuery

Pré-requisitos:

- Uma tabela do BigQuery que você deseja ler. Você precisará dos nomes da tabela e do conjunto de dados do BigQuery no formulário `[dataset].[table]`. Vamos chamar isso de *`tableName`*.
- O projeto de faturamento da tabela do BigQuery. Você precisará do nome do projeto, *`ParentProject`*. Se não houver um projeto principal de cobrança, use o projeto que contém a tabela.
- Informações de autenticação do BigQuery. Conclua as etapas para gerenciar suas credenciais de conexão com o AWS Glue para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *`ConnectionName`*.

Por exemplo:

```
bigquery_read = glueContext.create_dynamic_frame.from_options(
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "sourceType": "table",
        "table": "tableName",
    }
```

Você também pode fornecer uma consulta para filtrar os resultados retornados ao seu DynamicFrame. Você precisará configurar `query`, `sourceType`, `viewsEnabled` e `materializationDataset`.

Por exemplo:

Pré-requisitos adicionais

Você precisará criar ou identificar um conjunto de dados do BigQuery, *`MaterializationDataset, em que o BigQuery possa escrever visualizações materializadas`* para suas consultas.

Você precisará conceder as permissões apropriadas do GCP IAM à sua conta de serviço para criar tabelas no *`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"
            }
        )
```

Gravar em tabelas do BigQuery

Este exemplo grava diretamente no serviço BigQuery. O BigQuery também é compatível com o método de escrita “indireto”. Para obter mais informações sobre como configurar gravações indiretas, consulte the section called “Usar gravação indireta com o Google BigQuery”.

Pré-requisitos:

- Uma tabela do BigQuery na qual você gravar. Você precisará dos nomes da tabela e do conjunto de dados do BigQuery no formulário `[dataset].[table]`. Você também pode fornecer um novo nome de tabela que será criado automaticamente. Vamos chamar isso de *`tableName`*.
- O projeto de faturamento da tabela do BigQuery. Você precisará do nome do projeto, *`ParentProject`*. Se não houver um projeto principal de cobrança, use o projeto que contém a tabela.
- Informações de autenticação do BigQuery. Conclua as etapas para gerenciar suas credenciais de conexão com o AWS Glue para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *`ConnectionName`*.

Por exemplo:

```
bigquery_write = glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite,
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "writeMethod": "direct",
```

```
        "table": "tableName",
    }
)
```

Referência de opções de conexão BigQuery

- `project` — Padrão: conta de serviço padrão do Google Cloud. Usado para leitura/gravação. O nome de um projeto do Google Cloud associado à sua tabela.
- `table`: (obrigatório) usado para leitura/gravação. O nome da sua tabela do BigQuery no formato `[[project:]dataset.]`.
- `dataset` — Obrigatório quando não definido por meio da `table` opção. Usado para leitura/gravação. O nome do conjunto de dados que contém sua tabela do BigQuery.
- `parentProject` — Padrão: conta de serviço padrão do Google Cloud. Usado para leitura/gravação. O nome de um projeto do Google Cloud associado ao `project` usado para faturamento.
- `sourceType` - Usado para leitura. Obrigatório ao ler. Valores válidos: `table`, `query` informa ao AWS Glue se você lerá por tabela ou por consulta.
- `materializationDataset` - Usado para leitura. Valores válidos: strings. O nome de um conjunto de dados do BigQuery usado para armazenar materializações para visualizações.
- `viewsEnabled` - Usado para leitura. Padrão: falso. Valores válidos: true, false Configura se o BigQuery usará visualizações.
- `query` - Usado para leitura. Usado quando `viewsEnabled` for verdadeiro. Uma consulta DQL do GoogleSQL.
- `temporaryGcsBucket` — Usado para escrever. Obrigatório quando `writeMethod` estiver definido como padrão (`indirect`). Nome de um bucket do Google Cloud Storage usado para armazenar uma forma intermediária dos seus dados ao gravar no BigQuery.
- `writeMethod`: padrão: `indirect`. Valores válidos: `direct`, `indirect`. Usado para gravação. Especifica o método usado para gravar seus dados.
  - Se definido como `direct`, seu conector gravará usando a API BigQuery Storage Write.
  - Se definido como `indirect`, seu conector gravará no Google Cloud Storage e o transferirá para o BigQuery usando uma operação de carregamento. Sua conta de serviço do Google Cloud precisará das permissões apropriadas do GCS.

Usar gravação indireta com o Google BigQuery

Este exemplo usa gravação indireta, que grava dados no Google Cloud Storage e os copia para o Google BigQuery.

Pré-requisitos:

Você precisará de um bucket temporário do Google Cloud Storage, *TemporaryBucket*.

O papel do GCP IAM para a conta de serviço do GCP do AWS Glue precisará das permissões apropriadas do GCS para acessar o TemporaryBucket.

Configuração adicional

Para configurar a gravação indireta com o BigQuery:

1. Avalie the section called “Configurando o BigQuery” e localize ou baixe novamente seu arquivo JSON de credenciais do GCP. Identifique *secretName*, o AWS Secrets Manager segredo da conexão do Google BigQuery AWS Glue usada em seu trabalho.
2. Carregue seu arquivo JSON de credenciais em um local do Amazon S3 adequadamente seguro. Guarde o caminho para o arquivo, *s3secretpath, para futuras etapas*.
3. Edite *SecretName*, adicionando a `spark.hadoop.google.cloud.auth.service.account.json.keyfile` chave. Defina o valor como *s3secretpath*.
4. *Conceda ao seu trabalho do AWS Glue permissões do Amazon S3 IAM para acessar s3secretpath.*

Agora você pode fornecer a localização temporária do bucket do GCS ao seu método de gravação. Você não precisa fornecer `writeMethod`, pois historicamente `indirect` é o padrão.

```
bigquery_write = glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite,
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "temporaryGcsBucket": "temporaryBucket",
        "table": "tableName",
    }
)
```

Conexões JDBC

Certos tipos de banco de dados, geralmente relacionais, são compatíveis com conexão por meio do padrão JDBC. Para obter mais informações sobre o JDBC, consulte a documentação da Java JDBC API. O AWS Glue oferece suporte nativo à conexão a determinados bancos de dados por meio de conectores JDBC; as bibliotecas JDBC são fornecidas nos trabalhos do AWS Glue Spark. Ao se conectar a esses tipos de banco de dados usando as bibliotecas do AWS Glue, você tem acesso a um conjunto padrão de opções.

Os valores JDBCConnectionType incluem o seguinte:

- `"connectionType": "sqlserver"`: Designa uma conexão a um banco de dados do Microsoft SQL Server.
- `"connectionType": "mysql"`: designa uma conexão com um banco de dados MySQL.
- `"connectionType": "oracle"`: designa uma conexão com um banco de dados Oracle.
- `"connectionType": "postgresql"`: designa uma conexão com um banco de dados PostgreSQL.
- `"connectionType": "redshift"`: designa uma conexão com um banco de dados do Amazon Redshift. Para ter mais informações, consulte the section called “Conexões do Redshift”.

A tabela a seguir lista as versões do driver JDBC compatíveis com o AWS Glue.

Produto	Versões do driver de JDBC para o Glue 4.0	Versões do driver de JDBC para o Glue 3.0	Versões do driver de JDBC para o 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

Produto	Versões do driver de JDBC para o Glue 4.0	Versões do driver de JDBC para o Glue 3.0	Versões do driver de JDBC para o Glue 0.9, 1.0, 2.0
Amazon Redshift	redshift-jdbc42-2.1.0.16	redshift-jdbc41-1.2.12.1017	redshift-jdbc41-1.2.12.1017

* Para o tipo de conexão do Amazon Redshift, todas as outras opções de pares nome/valor que estão incluídas nas opções de conexão para uma conexão JDBC, incluindo opções de formatação, são passadas diretamente para a fonte de dados SparkSQL subjacente. Nos trabalhos do AWS Glue com Spark no AWS Glue 4.0 e versões posteriores, o conector nativo do AWS Glue para o Amazon Redshift usa a integração do Amazon Redshift para o Apache Spark. Para obter informações sobre como usar essas opções, consulte Integração do Amazon Redshift para o Apache Spark. Nas versões anteriores, consulte a fonte de dados do Amazon Redshift para Spark.

Para configurar a Amazon VPC para se conectar aos armazenamentos de dados do Amazon RDS usando JDBC, consulte the section called “Configurar um Amazon VPC para conectar aos armazenamentos de dados do Amazon RDS”.

> **Note**
>
> Os trabalhos do AWS são associados apenas a uma sub-rede durante uma execução. Isso pode afetar sua capacidade de se conectar a várias fontes de dados por meio do mesmo trabalho. Esse comportamento não se limita às fontes JDBC.

Tópicos

- Referência de opções de conexão JDBC
- Usar sampleQuery
- Usar o driver JDBC personalizado
- Leitura de tabelas JDBC em paralelo
- Configurar um Amazon VPC para conexões JDBC aos armazenamentos de dados do Amazon RDS desde o AWS Glue

Referência de opções de conexão JDBC

Se você já tiver uma conexão JDBC do AWS Glue definida, poderá reutilizar as propriedades de configuração nela definidas, como: url, usuário e senha, para não precisar especificá-las no código como opções de conexão. Esse atributo só está disponível no AWS Glue 3.0 e versões posteriores. Para fazer isso, use as seguintes propriedades de conexão:

- `"useConnectionProperties"`: defina como "true" para indicar que você deseja usar a configuração de uma conexão.
- `"connectionName"`: insira o nome da conexão da qual recuperar a configuração; a conexão deve ser definida na mesma região da tarefa.

Use estas opções de conexão com conexões JDBC:

- `"url"`: (Obrigatório) O URL do JDBC para o banco de dados.
- `"dbtable"`: (obrigatório) o banco de dados a ser lido. Para armazenamentos de dados JDBC que oferecem suporte a esquemas dentro de um banco de dados, especifique `schema.table-name`. Se um esquema não for fornecido, o esquema "público" padrão será usado.
- `"user"`: (obrigatório) o nome de usuário a ser usado ao se conectar.
- `"password"`: (Obrigatório) A senha a ser usada ao se conectar.
- (Opcional) As opções a seguir permitem que você forneça um driver do JDBC personalizado. Use estas opções se for necessário usar um driver que não é compatível nativamente com o AWS Glue.

  Os trabalhos de ETL podem usar diferentes versões de driver JDBC para o destino e a fonte de dados, mesmo que o destino e a origem sejam o mesmo produto de banco de dados. Isso permite migrar dados entre bancos de dados de fonte e de destino com versões diferentes. Para usar estas opções, é necessário primeiro fazer upload do arquivo JAR do driver do JDBC para o Amazon S3.

  - `"customJdbcDriverS3Path"`: o caminho no Amazon S3 do driver JDBC personalizado.
  - `"customJdbcDriverClassName"`: o nome da classe do driver JDBC.
- `"bulkSize"`: (opcional) usado para configurar inserções paralelas para acelerar cargas em massa em destinos de JDBC. Especifique um valor inteiro para o grau de paralelismo a ser usado ao gravar ou inserir dados. Essa opção é útil para melhorar a performance de gravações em bancos de dados, como o Arch User Repository (AUR).

- `"hashfield"` (Opcional) Uma string, usada para especificar o nome de uma coluna na tabela JDBC a ser usada para dividir os dados em partições ao ler tabelas JDBC em paralelo. Forneça "hashfield" OU "hashexpression". Para ter mais informações, consulte the section called “Leitura do JDBC em paralelo”.
- `"hashexpression"` (Opcional) Uma cláusula de seleção SQL retornando um número inteiro. Usada para dividir os dados em uma tabela JDBC em partições ao ler tabelas JDBC em paralelo. Forneça "hashfield" OU "hashexpression". Para ter mais informações, consulte the section called “Leitura do JDBC em paralelo”.
- `"hashpartitions"` (Opcional) Um número inteiro positivo. Usado para especificar o número de leituras paralelas da tabela JDBC ao ler tabelas JDBC em paralelo. Padrão: 7. Para ter mais informações, consulte the section called “Leitura do JDBC em paralelo”.
- `"sampleQuery"`: (opcional) uma instrução de consulta SQL personalizada. Usada para especificar um subconjunto de informações em uma tabela para recuperar uma amostra do conteúdo da tabela. Quando configurada sem considerar os dados, pode ser menos eficiente do que os métodos DynamicFrame, causando timeouts ou erros de falta de memória. Para ter mais informações, consulte the section called “Usar sampleQuery”.
- `"enablePartitioningForSampleQuery"`: (opcional) um booleano. Padrão: falso. Usado para permitir a leitura de tabelas JDBC em paralelo ao especificar `sampleQuery`. Se definido como verdadeiro, **`sampleQuery`** deve terminar com "where" ou "and" para o AWS Glue para anexar condições de particionamento. Para ter mais informações, consulte the section called “Usar sampleQuery”.
- `"sampleSize"`: (opcional) um número inteiro positivo. Limita o número de linhas retornado pela consulta de amostra. Funciona somente quando `enablePartitioningForSampleQuery` é verdadeiro. Se o particionamento não estiver habilitado, você deverá adicionar diretamente `"limit x"` à `sampleQuery` para limitar o tamanho. Para ter mais informações, consulte the section called “Usar sampleQuery”.

Usar sampleQuery

Esta seção explica como usar `sampleQuery`, `sampleSize` e `enablePartitioningForSampleQuery`.

`sampleQuery` pode ser uma forma eficiente de amostrar algumas linhas do conjunto de dados. Por padrão, a consulta é executada por um único executor. Quando configurada sem considerar os dados, pode ser menos eficiente do que os métodos DynamicFrame, causando timeouts ou erros de falta de memória. A execução de SQL no banco de dados subjacente como parte do pipeline de ETL

geralmente só é necessária para fins de performance. Se você estiver tentando visualizar algumas linhas do conjunto de dados, considere usar the section called “show”. Se você estiver tentando transformar o conjunto de dados usando SQL, considere usar the section called “toDF” para definir uma transformação de SparkSQL para os dados no formato DataFrame.

Embora sua consulta possa manipular uma variedade de tabelas, a `dbtable` continua sendo obrigatória.

Usar sampleQuery para recuperar uma amostra da tabela

Ao usar o comportamento padrão de sampleQuery para recuperar uma amostra dos dados, o AWS Glue não espera um throughput substancial, assim ele executa a consulta em um único executor. Para limitar os dados que você fornece e não causar problemas de performance, sugerimos que você inclua uma cláusula `LIMIT` no SQL.

Example Usar SampleQuery sem particionamento

O exemplo de código a seguir mostra como usar `sampleQuery` sem particionamento.

```
//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()
```

Usar sampleQuery em conjuntos de dados maiores

Se você estiver lendo um conjunto de dados grande, talvez seja necessário ativar o particionamento JDBC para consultar uma tabela em paralelo. Para ter mais informações, consulte the section called “Leitura do JDBC em paralelo”. Para usar `sampleQuery` com particionamento JDBC, defina `enablePartitioningForSampleQuery` como verdadeiro. A ativação desse atributo exige que você faça algumas alterações na `sampleQuery`.

Ao usar particionamento JDBC com `sampleQuery`, a consulta deve terminar com "where" ou "and" para o AWS Glue para anexar condições de particionamento.

Se você quiser limitar os resultados da sampleQuery ao ler tabelas JDBC em paralelo, defina o parâmetro `"sampleSize"` em vez de especificar uma cláusula `LIMIT`.

Example Usar SampleQuery com particionamento JDBC

O exemplo de código a seguir mostra como usar `sampleQuery` com particionamento 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()
```

Regras e restrições:

Consultas de amostra não podem ser usadas junto com marcadores de trabalho. O estado do marcador será ignorado quando a configuração de ambos for fornecida.

Usar o driver JDBC personalizado

Os exemplos de código a seguir mostram como fazer a leitura e gravação de bancos de dados JDBC com drivers JDBC personalizados. Eles demonstram a leitura de uma versão de um produto de banco de dados e a gravação em uma versão posterior do mesmo produto.

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)
  }
}
```

Leitura de tabelas JDBC em paralelo

Você pode definir propriedades da sua tabela JDBC para permitir que o AWS Glue leia dados em paralelo. Ao definir determinadas propriedades, você instrui o AWS Glue a executar consultas SQL paralelas em partições lógicas dos seus dados. Você pode controlar o particionamento definindo um campo de hash ou uma expressão de hash. Você também pode controlar o número de leituras paralelas que são usadas para acessar seus dados.

Ler as tabelas JDBC em paralelo é uma técnica de otimização que pode melhorar a performance. Para obter mais informações sobre o processo para identificar quando essa técnica é adequada, consulte Reduce the amount of data scan no guia Best practices for performance tuning AWS Glue for Apache Spark jobs em AWS Prescriptive Guidance.

Para habilitar leituras paralelas, você pode definir pares de chave e valor no campo de parâmetros da estrutura da tabela. Use a notação JSON para definir um valor para o campo de parâmetro da tabela. Para obter mais informações sobre como editar as propriedades de uma tabela, consulte Exibir e editar detalhes da tabela. Você também pode habilitar leituras paralelas ao chamar os métodos de ETL (extração, transformação e carregamento) `create_dynamic_frame_from_options` e `create_dynamic_frame_from_catalog`. Para obter mais informações sobre como especificar opções nesses métodos, consulte from_options e from_catalog.

Você pode usar esse método para tabelas JDBC, ou seja, a maioria das tabelas cujos dados de base são um armazenamento de dados JDBC. Essas propriedades são ignoradas durante a leitura de tabelas do Amazon RedShift e Amazon S3.

hashfield

> Defina `hashfield` como o nome de uma coluna na tabela JDBC a ser usada para dividir os dados em partições. Para obter os melhores resultados, essa coluna deve ter uma distribuição uniforme de valores para distribuir os dados entre partições. Essa coluna pode ser de qualquer tipo de dados. O AWS Glue gera consultas não sobrepostas que são executadas em paralelo para ler os dados particionados por essa coluna. Por exemplo, se os dados estiverem distribuídos uniformemente por mês, você poderá usar a coluna `month` para ler cada mês de dados em paralelo.
>
> ```
> 'hashfield': 'month'
> ```

O AWS Glue cria uma consulta para definir o hash do valor do campo como um número de partição e executa essa consulta para todas as partições em paralelo. Para usar sua própria consulta para particionar uma leitura de tabela, forneça uma tabela `hashexpression` em vez de um `hashfield`.

hashexpression

Defina `hashexpression` como uma expressão SQL (de acordo com a gramática do mecanismo de banco de dados JDBC) que retorne um número inteiro. Uma expressão simples é o nome de qualquer coluna numérica na tabela. O AWS Glue gera consultas SQL para ler os dados JDBC em paralelo usando o `hashexpression` na cláusula `WHERE` para particionar os dados.

Por exemplo, use a coluna numérica `customerID` para ler dados particionados por um número de cliente.

```
'hashexpression': 'customerID'
```

Para que o AWS Glue controle o particionamento, forneça um `hashfield` em vez de um `hashexpression`.

hashpartitions

Defina `hashpartitions` como o número de leituras paralelas da tabela JDBC. Se essa propriedade não for definida, o valor padrão será 7.

Por exemplo, defina o número de leituras em paralelo como 5 para que o AWS Glue lê os dados com cinco consultas (ou menos).

```
'hashpartitions': '5'
```

Configurar um Amazon VPC para conexões JDBC aos armazenamentos de dados do Amazon RDS desde o AWS Glue

Ao usar o JDBC para conectar a bancos de dados no Amazon RDS, você precisará realizar configurações adicionais. Para permitir que os componentes do AWS Glue se comuniquem com o Amazon RDS, é necessário configurar o acesso aos seus armazenamentos de dados do Amazon RDS no Amazon VPC. Para permitir que o AWS Glue se comunique entre seus componentes,

especifique um grupo de segurança com uma regra de entrada de autorreferência para todas as portas TCP. Ao criar uma regra de autorreferência, você pode restringir a origem ao mesmo security group na VPC. Uma regra de autorreferência não abrirá a VPC para todas as redes. O security group padrão para sua VPC pode já conter uma regra de entrada de autorreferenciada para ALL Traffic.

Para configurar o acesso entre os armazenamentos de dados do AWS Glue e do Amazon RDS

1. Faça login AWS Management Console e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. No console do Amazon RDS, identifique os grupos de segurança usados para controlar o acesso ao banco de dados do Amazon RDS.

   No painel de navegação esquerdo, escolha Bancos de dados e, em seguida, selecione a instância à qual você gostaria de se conectar na lista no painel principal.

   Na página de detalhes do banco de dados, encontre Grupos de segurança da VPC na guia Conectividade e segurança.
3. Com base na sua arquitetura de rede, identifique qual grupo de segurança associado é melhor modificar para permitir o acesso ao serviço AWS Glue. Salve seu nome, *database-security-group*para futura referência. Se não houver um grupo de segurança apropriado, siga as instruções para Fornecer acesso à instância de banco de dados na VPC criando um grupo de segurança na documentação do Amazon RDS.
4. Faça login AWS Management Console e abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
5. No console da Amazon VPC, identifique como atualizar. *database-security-group*

   No painel de navegação esquerdo, escolha Grupos de segurança e *database-security-group*selecione na lista no painel principal.
6. Identifique o ID do grupo de segurança para *database-security-group*, *database-sg-id*. Guarde-o para uso posterior.

   Na página de detalhes do grupo de segurança, encontre ID do grupo de segurança.
7. Altere as regras de entrada para *database-security-group*, adicione uma regra de autorreferência para permitir que os AWS Glue componentes se comuniquem. Especificamente, adicione ou confirme se há uma regra em que Tipo é`All TCP`, Protocolo é`TCP`, Intervalo de Portas inclui todas as portas e Origem é *database-sg-id*. Verifique se o grupo de segurança que você inseriu para Origem é o mesmo que o grupo de segurança que você está editando.

Na página de detalhes do grupo de segurança, selecione Editar regras de entrada.

A regra de entrada é semelhante a esta:

Tipo	Protocolo	Intervalo de portas	Origem
Todos os TCP	TCP	0–65535	*database-sg-id*

8. Adicione regras para o tráfego de saída.

Na página de detalhes do grupo de segurança, selecione Editar regras de saída.

Se o grupo de segurança permitir todo o tráfego de saída, você não precisará de regras separadas. Por exemplo: .

Tipo	Protocolo	Intervalo de portas	Destino
Todo o tráfego	ALL	ALL	0.0.0.0/0

Se sua arquitetura de rede foi desenvolvida para que você restrinja o tráfego de saída, crie as seguintes regras de saída:

Crie uma regra de autorreferência em que Tipo é`All TCP`, Protocolo é`TCP`, Intervalo de Portas inclui todas as portas e Destino é. *database-sg-id* Verifique se o grupo de segurança que você inseriu para Destino é o mesmo que o grupo de segurança que você está editando.

Se você estiver usando um endpoint da VPC do Amazon S3, adicione uma regra de HTTPS para permitir o tráfego da VPC para o Amazon S3. *Crie uma regra em que* ***Tipo*** *sejaHTTPS,* ***Protocolo*** *sejaTCP,* ***Intervalo de Portas*** *seja 443 e* ***Destino*** *seja o ID da lista de prefixos gerenciados para o endpoint do gateway Amazon S3, s3-. prefix-list-id* Para obter mais informações sobre listas de prefixos e endpoints de gateway do Amazon S3, consulte Endpoints de gateway para Amazon S3 na documentação do Amazon VPC.

Por exemplo: .

Tipo	Protocolo	Intervalo de portas	Destino
Todos os TCP	TCP	0–65535	*database-sg-id*
HTTPS	TCP	443	*s3- prefix-list-id*

Conexões do MongoDB

O AWS Glue para Spark pode ser usado para ler e gravar em tabelas no MongoDB e MongoDB Atlas no AWS Glue 4.0 e versões posteriores. Conecte-se ao MongoDB usando credenciais de nome de usuário e senha armazenadas no AWS Secrets Manager via conexão do AWS Glue.

Para obter mais informações sobre o MongoDB, consulte a Documentação do MongoDB.

Configurar conexões do MongoDB

Para se conectar ao MongoDB via AWS Glue, você precisará de suas credenciais do MongoDB, *mongodbUser* e *mongodbPass*.

Para se conectar ao MongoDB via AWS Glue, talvez seja necessário atender a alguns pré-requisitos:

- Se a sua instância do MongoDB estiver em uma Amazon VPC, configure a Amazon VPC para permitir que seu trabalho do AWS Glue se comunique com a instância do MongoDB sem que o tráfego passe pela Internet pública.

  Na Amazon VPC, identifique ou crie uma VPC, uma Sub-rede e um Grupo de segurança que o AWS Glue usará durante a execução do trabalho. Além disso, você precisa garantir que a Amazon VPC esteja configurada para permitir o tráfego de rede entre sua instância do MongoDB e esse local. Com base no layout da rede, isso pode exigir alterações em regras do grupo de segurança, ACLs de rede, gateways de NAT e conexões de emparelhamento.

Em seguida, você pode continuar com a configuração do AWS Glue para usá-lo com o MongoDB.

Para configurar uma conexão com o MongoDB:

1. Opcionalmente, no AWS Secrets Manager, crie um segredo usando suas credenciais do MongoDB. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma

AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *SecretName*, para a próxima etapa.

- Ao selecionar Pares chave/valor, crie um par para a chave `username` com o valor *mongodbUser*.

  Ao selecionar Pares chave/valor, crie um par para a chave `password` com o valor *mongodbPass*.

2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *connectionName*, para uso futuro no AWS Glue.

   - Ao selecionar um Tipo de conexão, selecione MongoDB ou MongoDB Atlas.
   - Ao selecionar o URL do MongoDB ou o URL do MongoDB Atlas, forneça o nome de host da sua instância do MongoDB.

     Um URL do MongoDB é fornecido no formato `mongodb://`*mongoHost*`:`*mongoPort*`/`*mongoDBname*.

     Um URL do MongoDB Atlas é fornecido no formato `mongodb+srv://`*mongoHost*`:`*mongoPort*`/`*mongoDBname*.

     Fornecer o banco de dados padrão *mongoDBname* para a conexão é opcional.
   - Se você optar por criar um segredo do Secrets Manager, escolha o Tipo de credencial do AWS Secrets Manager.

     Em seguida, em Segredo do AWS, forneça o *secretName*.
   - *Se você optar por fornecer* ***Nome de usuário e senha****, forneça também o mongodbUser e o mongodbPass*.

3. Nas seguintes situações, configurações adicionais podem ser necessárias:

   - Para instâncias do MongoDB hospedadas na AWS em uma Amazon VPC
     - Será necessário fornecer informações de conexão da Amazon VPC à conexão do AWS Glue que define suas credenciais de segurança do MongoDB. Ao criar ou atualizar sua conexão, defina VPC, Sub-rede e Grupos de segurança em Opções de rede.

Depois de criar uma conexão AWS Glue MongoDB, siga estas etapas a seguir antes de chamar seu método de conexão.

- Se você optar por criar um segredo do Secrets Manager, conceda ao perfil do IAM associado ao seu trabalho do AWS Glue permissão para ler *secretName*.
- Na configuração do trabalho do AWS Glue, forneça *connectionName* como uma conexão de rede adicional.

Para usar sua conexão AWS Glue MongoDB no AWS Glue para Spark, forneça a opção `connectionName` na chamada ao método de conexão. Como alternativa, você pode seguir as etapas the section called “Integração com MongoDB” para usar a conexão em conjunto com o AWS Glue Data Catalog.

Ler do MongoDB usando uma conexão do AWS Glue

Pré-requisitos:

- Uma coleção do MongoDB da qual você gostaria de ler. Você precisará de informações de identificação para a coleção.

  Uma coleção do MongoDB é identificada por um nome de banco de dados e um nome de coleção, *mongodbName* e *mongodbCollection*.
- Uma conexão AWS Glue MongoDB configurada para fornecer informações de autenticação. Conclua as etapas do procedimento anterior, Para configurar uma conexão com o MongoDB para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Por exemplo:

```
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",
    }
)
```

Escrever em tabelas do MongoDB

Este exemplo escreve informações de um DynamicFrame existente, *dynamicFrame*, no MongoDB.

Pré-requisitos:

- Uma coleção do MongoDB na qual você gostaria de escrever. Você precisará de informações de identificação para a coleção.

  Uma coleção do MongoDB é identificada por um nome de banco de dados e um nome de coleção, *mongodbName* e *mongodbCollection*.
- Uma conexão AWS Glue MongoDB configurada para fornecer informações de autenticação. Conclua as etapas do procedimento anterior, Para configurar uma conexão com o MongoDB para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Por exemplo:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="mongodb",
    connection_options={
        "connectionName": "connectionName",
        "database": "mongodbName",
        "collection": "mongodbCollection",
        "disableUpdateUri": "false",
        "retryWrites": "false",
    },
)
```

Leitura e escrita em tabelas no MongoDB

Este exemplo escreve informações de um DynamicFrame existente, *dynamicFrame*, no MongoDB.

Pré-requisitos:

- Uma coleção do MongoDB da qual você gostaria de ler. Você precisará de informações de identificação para a coleção.

  Uma coleção do MongoDB na qual você gostaria de escrever. Você precisará de informações de identificação para a coleção.

  Uma coleção do MongoDB é identificada por um nome de banco de dados e um nome de coleção, *mongodbName* e *mongodbCollection*.
- Informações de autenticação do MongoDB, *mongodbUser* e *mongodbPassword*.

Por exemplo:

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)
  }
}
```

Referência de opções de conexão do MongoDB

Designa uma conexão com o MongoDB. As opções de conexão diferem para uma conexão da fonte e uma conexão do coletor.

Essas propriedades de conexão são compartilhadas entre as conexões de origem e coletor:

- `connectionName`: usado para leitura/gravação. O nome de uma conexão do AWS Glue MongoDB configurada para fornecer informações de autenticação e rede ao seu método de conexão. Quando uma conexão do AWS Glue é configurada conforme descrito na seção the section called “Configurar o MongoDB” anterior, fornecer `connectionName` substituirá a necessidade de fornecer as opções de conexão `"uri"`, `"username"` e `"password"`.
- `"uri"`: (obrigatório) o host do MongoDB do qual fazer a leitura, formatado como `mongodb://<host>:<port>`. Usado nas versões do AWS Glue anteriores ao AWS Glue 4.0.
- `"connection.uri"`: (obrigatório) o host do MongoDB do qual fazer a leitura, formatado como `mongodb://<host>:<port>`. Usado no AWS Glue 4.0 e versões posteriores.
- `"username"`: (obrigatório) o nome de usuário do MongoDB.
- `"password"`: (obrigatório) a senha do MongoDB.
- `"database"`: (obrigatório) o banco de dados MongoDB do qual fazer a leitura.
  Essa opção também pode ser passada em `additional_options` ao chamar `glue_context.create_dynamic_frame_from_catalog` em seu script de trabalho.
- `"collection"`: (obrigatório) a coleção do MongoDB da qual fazer a leitura.
  Essa opção também pode ser passada em `additional_options` ao chamar `glue_context.create_dynamic_frame_from_catalog` em seu script de trabalho.

“connectionType”: “mongodb” como fonte

Use as seguintes opções de conexão com `"connectionType": "mongodb"` como fonte:

- `"ssl"`: (opcional) se for `true`, iniciará uma conexão SSL. O padrão é `false`.
- `"ssl.domain_match"`: (opcional) se for `true` e se `ssl` for `true`, será realizada a verificação de correspondência do domínio. O padrão é `true`.
- `"batchSize"`: (opcional) o número de documentos que devem ser retornados por lote, usado no cursor de lotes internos.
- `"partitioner"`: (opcional) o nome da classe do particionador do qual fazer a leitura de dados de entrada do MongoDB. O conector fornece os seguintes particionadores:
  - `MongoDefaultPartitioner` (padrão) (sem suporte no AWS Glue 4.0)
  - `MongoSamplePartitioner` (requer o MongoDB 3.2 ou posterior) (Sem suporte no AWS Glue 4.0)
  - `MongoShardedPartitioner` (sem suporte no AWS Glue 4.0)

  - `MongoSplitVectorPartitioner` (sem suporte no AWS Glue 4.0)
  - `MongoPaginateByCountPartitioner` (sem suporte no AWS Glue 4.0)
  - `MongoPaginateBySizePartitioner` (sem suporte no 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) opções para o particionador designado. As seguintes opções são compatíveis com cada particionador:
  - `MongoSamplePartitioner`: `partitionKey`, `partitionSizeMB`, `samplesPerPartition`
  - `MongoShardedPartitioner`: `shardkey`
  - `MongoSplitVectorPartitioner`: `partitionKey`, `partitionSizeMB`
  - `MongoPaginateByCountPartitioner`: `partitionKey`, `numberOfPartitions`
  - `MongoPaginateBySizePartitioner`: `partitionKey`, `partitionSizeMB`

  Para obter mais informações sobre essas opções, consulte Partitioner Configuration na documentação do MongoDB.

“connectionType”: “mongodb” como coletor

Use as seguintes opções de conexão com `"connectionType": "mongodb"` como coletor:

- `"ssl"`: (opcional) se for `true`, iniciará uma conexão SSL. O padrão é `false`.
- `"ssl.domain_match"`: (opcional) se for `true` e se `ssl` for `true`, será realizada a verificação de correspondência do domínio. O padrão é `true`.
- `"extendedBsonTypes"`: (opcional) se for `true`, permite tipos de BSON estendidos ao gravar dados no MongoDB. O padrão é `true`.
- `"replaceDocument"`: (opcional) se for `true`, substituirá todo o documento ao salvar conjuntos de dados que contêm um campo `_id`. Se for `false`, serão atualizados somente os campos do documento que correspondam aos campos do conjunto de dados. O padrão é `true`.
- `"maxBatchSize"`: (opcional) o tamanho máximo do lote para operações em massa ao salvar dados. O padrão é 512.
- `"retryWrites"`: (Opcional): repita automaticamente determinadas operações de escrita uma única vez se o AWS Glue encontrar um erro de rede.

Conexões do SAP HANA

O AWS Glue para Spark pode ser usado para ler e escrever em tabelas no SAP HANA no AWS Glue 4.0 e versões posteriores. Você pode definir o que ler no SAP HANA com uma consulta SQL. Conecte-se ao SAP HANA usando credenciais JDBC armazenadas no AWS Secrets Manager via conexão AWS Glue SAP HANA.

Para obter mais informações sobre o JDBC do SAP HANA, consulte a Documentação do SAP HANA.

Configurar conexões do SAP HANA

Para se conectar ao SAP HANA via AWS Glue, será necessário criar e armazenar suas credenciais do SAP HANA em um segredo do AWS Secrets Manager e, em seguida, associar esse segredo a uma conexão ao AWS Glue do SAP HANA. Você precisará configurar a conectividade de rede entre seu serviço SAP HANA e o AWS Glue.

Para se conectar ao SAP HANA, talvez seja necessário atender a alguns pré-requisitos:

- Se o seu serviço SAP HANA estiver em uma Amazon VPC, configure a Amazon VPC para permitir que seu trabalho do AWS Glue se comunique com o serviço SAP HANA sem que o tráfego passe pela Internet pública.

  Na Amazon VPC, identifique ou crie uma VPC, uma Sub-rede e um Grupo de segurança que o AWS Glue usará durante a execução do trabalho. Além disso, você precisa garantir que a Amazon VPC esteja configurada para permitir o tráfego de rede entre seu endpoint SAP HANA e esse local. Seu trabalho precisará estabelecer uma conexão TCP com a porta JDBC do SAP HANA. Para obter mais informações sobre as portas do SAP HANA, consulte a Documentação do SAP HANA. Com base no layout da rede, isso pode exigir alterações em regras do grupo de segurança, ACLs de rede, gateways de NAT e conexões de emparelhamento.
- Não há pré-requisitos adicionais quando seu endpoint do SAP HANA está acessível pela Internet.

Para configurar uma conexão com o SAP HANA:

1. No AWS Secrets Manager, crie um segredo usando suas credenciais do SAP HANA. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *`SecretName`*, para a próxima etapa.

- Ao selecionar Pares chave/valor, crie um par para a chave `user` com o valor *`saphanaUsername`*.
- Ao selecionar Pares chave/valor, crie um par para a chave `password` com o valor *`saphanaPassword`*.

2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *`connectionName`*, para uso futuro no AWS Glue.

   - Ao selecionar um Tipo de conexão, selecione SAP HANA.
   - Ao fornecer o URL do SAP HANA, forneça o URL da sua instância.

     Os URLs JDBC do SAP HANA estão no formato `jdbc:sap://`*`saphanaHostname`*`:`*`saphanaPort`*`/?`*`databaseName`*`=`*`saphanaDBname`*`,`*`Paramete`*

     O AWS Glue requer os seguintes parâmetros de URL do JDBC:

     - `databaseName`: um banco de dados padrão no SAP HANA ao qual se conectar.
   - Ao selecionar um Segredo da AWS, forneça o *`secretName`*.

Depois de criar uma conexão AWS Glue SAP HANA, siga estas etapas antes de executar seu trabalho do AWS Glue:

- Conceda ao perfil do IAM associada ao seu trabalho do AWS Glue permissão para ler *`secretName`*.
- Na configuração do trabalho do AWS Glue, forneça *`connectionName`* como uma conexão de rede adicional.

Ler de tabelas do SAP HANA

Pré-requisitos:

- Uma tabela do SAP HANA da qual você deseja ler. Você precisará de informações de identificação para a tabela.

  Uma tabela pode ser especificada com um nome de tabela e um nome de esquema do SAP HANA, no formulário *`schemaName`*`.`*`tableName`*. O nome do esquema e o separador "." não serão necessários se a tabela estiver no esquema padrão, "público". Chame isso de

tableIdentifier. Observe que o banco de dados é fornecido como um parâmetro de URL do JDBC em `connectionName`.

- Uma conexão AWS Glue SAP HANA configurada para fornecer informações de autenticação. Conclua as etapas do procedimento anterior, Para configurar uma conexão com o SAP HANA para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Por exemplo:

```
saphana_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="saphana",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier",
    }
)
```

Você também pode fornecer uma consulta SQL SELECT para filtrar os resultados retornados ao seu DynamicFrame. Você precisará configurar `query`.

Por exemplo:

```
saphana_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="saphana",
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

Escrever em tabelas do SAP HANA

Este exemplo escreve informações de um DynamicFrame existente, *dynamicFrame*, no SAP HANA. Se a tabela já contiver informações, o AWS Glue irá gerar um erro.

Pré-requisitos:

- Uma tabela do SAP HANA na qual você gostaria de escrever.

Uma tabela pode ser especificada com um nome de tabela e um nome de esquema do SAP HANA, no formulário *schemaName*.*tableName*. O nome do esquema e o separador "." não serão necessários se a tabela estiver no esquema padrão, "público". Chame isso de *tableIdentifier*. Observe que o banco de dados é fornecido como um parâmetro de URL do JDBC em `connectionName`.

- Informações de autenticação do SAP HANA. Conclua as etapas do procedimento anterior, Para configurar uma conexão com o SAP HANA para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Por exemplo:

```
options = {
    "connectionName": "connectionName",
    "dbtable": 'tableIdentifier'
}

    saphana_write = glueContext.write_dynamic_frame.from_options(
        frame=dynamicFrame,
        connection_type="saphana",
        connection_options=options
)
```

Referência de opções de conexão do SAP HANA

- `connectionName` — Obrigatório. Usado para leitura/gravação. O nome de uma conexão do AWS Glue SAP HANA configurada para fornecer informações de autenticação e rede ao seu método de conexão.
- `databaseName`: usado para leitura/gravação. Valores válidos: nomes de bancos de dados no SAP HANA. O nome do banco de dados ao qual se conectar.
- `dbtable`: necessário para escrever, obrigatório para leitura, a menos que `query` seja fornecido. Usado para leitura/gravação. Valores válidos: conteúdo de uma cláusula SQL FROM do SAP HANA. Identifica uma tabela no SAP HANA à qual se conectar. Você também pode fornecer outro SQL além do nome de uma tabela, como uma subconsulta. Para obter mais informações, consulte a cláusula From na documentação do SAP HANA.
- `query` - Usado para leitura. Uma consulta SQL SELECT do SAP HANA que define o que deve ser recuperado na leitura do SAP HANA.

Conexões do Snowflake

Você pode usar o AWS Glue para Spark para ler e gravar em tabelas no Snowflake no AWS Glue 4.0 e versões posteriores. Você pode ler no Snowflake com uma consulta SQL. Você pode se conectar ao Snowflake usando um usuário e uma senha. Você pode consultar as credenciais do Snowflake armazenadas no AWS Secrets Manager por meio do catálogo de dados do AWS Glue . As credenciais do Snowflake do catálogo de dados do AWS Glue para Spark são armazenadas separadamente das credenciais do Snowflake do catálogo de dados para crawlers. Você deve escolher um tipo de conexão do `SNOWFLAKE` e não um tipo de conexão `JDBC` configurada para se conectar ao Snowflake.

Para obter mais informações sobre o Snowflake, consulte o site do Snowflake. Para obter mais informações sobre o Snowflake na AWS, consulte Snowflake Data Warehouse na Amazon Web Services.

Configurar conexões do Snowflake

Não há pré-requisitos da AWS para se conectar aos fluxos do Snowflake disponíveis pela Internet.

Opcionalmente, você pode realizar a seguinte configuração para gerenciar suas credenciais de conexão com o AWS Glue.

Para gerenciar suas credenciais de conexão com AWS o Glue

1. No Snowflake, gere um usuário, *`SnowflakeUser`* e senha, *`snowflakePassword`*.
2. Nos AWS Secrets Manager, crie um segredo usando suas credenciais do Snowflake. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criação de um segredo do AWS Secrets Manager na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *`SecretName`*, para a próxima etapa.
   - Ao selecionar pares de chave/valor, crie um par para *`snowflakeUser`* com a chave `sfUser`.
   - Ao selecionar pares de chave/valor, crie um par para *`snowflakePassword`* com a chave `sfPassword`.
   - Ao selecionar pares de chave/valor, você pode fornecer a chave ao seu armazém do Snowflake `sfWarehouse`.
3. No catálogo de dados do AWS Glue, crie uma conexão seguindo as etapas em the section called “Adicionar uma conexão do AWS Glue”. Depois de criar a conexão, guarde o nome da conexão, *`connectionName`*, para a próxima etapa.

- Ao selecionar um Tipo de conexão, selecione Snowflake.
- Ao selecionar a URL do Snowflake, forneça a URL da sua instância do Snowflake. A URL usará um nome de host no formulário *account_identifier*.snowflakecomputing.com.
- Ao selecionar um Segredo da AWS, forneça o *secretName*.

4. Na configuração do trabalho do AWS Glue, forneça *connectionName* como uma conexão de rede adicional.

Nas seguintes situações, você pode precisar do seguinte:

- Para Snowflake hospedado na AWS em uma Amazon VPC
  - Você precisará da configuração apropriada da Amazon VPC para o Snowflake. Para obter mais informações sobre como configurar a Amazon VPC, consulte AWS PrivateLink & Snowflake na documentação do Snowflake.
  - Você precisará da configuração apropriada da Amazon VPC para o AWS Glue. the section called “Endpoint da VPC (AWS PrivateLink)”.
  - Você precisará criar uma conexão do catálogo de dados do AWS Glue que forneça informações de conexão do Amazon VPC (além da identificação de um AWS Secrets Manager segredo que define suas credenciais de segurança do Snowflake). A URL mudará quando o AWS PrivateLink estiver sendo usado, conforme descrito na documentação do Snowflake associada a um item anterior.
  - Você precisará que a configuração do trabalho inclua a conexão do catálogo de dados como uma Conexão de rede adicional.

Ler das tabelas do Snowflake

Pré-requisitos: uma tabela do Snowflake que você gostaria de ler. *Você precisará do nome da tabela Snowflake, tableName*. Você precisará da URL do Snowflake, *snowflakeUrl*, do nome de usuário *snowflakeUser* e da senha *snowflakePassword*. *Se o usuário do Snowflake não tiver um namespace padrão definido, você precisará do nome do banco de dados do Snowflake databaseName e do nome do esquema schemaName*. *Além disso, se o usuário do Snowflake não tiver um armazém padrão definido, você precisará de um nome de armazém warehouseName*.

Por exemplo:

Pré-requisitos adicionais: conclua as etapas de Para gerenciar suas credenciais de conexão com o AWS Glue para configurar *snowflakeURL*, *snowflakeUsername* e *snowflakePassword*. Para revisar essas etapas, consulte the section called “Configurar o Snowflake” na seção anterior. Para selecionar com qual conexão de rede adicional se conectar, usaremos o parâmetro `connectionName`.

```
snowflake_read = glueContext.create_dynamic_frame.from_options(
  connection_type="snowflake",
  connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
    }
)
```

Além disso, você pode usar os parâmetros `autopushdown` e `query` para ler uma parte de uma tabela do Snowflake. Isso pode ser substancialmente mais eficiente do que filtrar os resultados depois de serem carregados no Spark. Considere um exemplo em que todas as vendas são armazenadas na mesma tabela, mas você só precisa analisar as vendas de uma determinada loja nos feriados. Se essas informações estiverem armazenadas na tabela, você poderá usar pushdown de predicado para recuperar os resultados da seguinte forma:

```
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",
    }
)
```

Gravar em tabelas do Snowflake

Pré-requisitos: um banco de dados do Snowflake no qual você deseja gravar. Você precisará de um nome de tabela atual ou desejado, *tableName*. Você precisará da URL do Snowflake, *snowflakeUrl*, do nome de usuário *snowflakeUser* e da senha *snowflakePassword*. *Se o*

usuário do Snowflake não tiver um namespace padrão definido, você precisará do nome do banco de dados do Snowflake databaseName e do nome do esquema schemaName. Além disso, se o usuário do Snowflake não tiver um armazém padrão definido, você precisará de um nome de armazém warehouseName.

Por exemplo:

Pré-requisitos adicionais: conclua as etapas de Para gerenciar suas credenciais de conexão com o AWS Glue para configurar *snowflakeURL*, *snowflakeUsername* e *snowflakePassword*. Para revisar essas etapas, consulte the section called “Configurar o Snowflake” na seção anterior. Para selecionar com qual conexão de rede adicional se conectar, usaremos o parâmetro `connectionName`.

```
glueContext.write_dynamic_frame.from_options(
    connection_type="snowflake",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
    },
)
```

Referência de opções de conexão do Snowflake

O tipo de conexão do Snowflake usa as seguintes opções de conexão:

Você pode recuperar alguns dos parâmetros desta seção de uma conexão do catálogo de dados (`sfUrl`, `sfUser`, `sfPassword`) e, nesse caso, não será necessário fornecê-los. É possível fazer isso fornecendo o parâmetro `connectionName`.

Você pode recuperar alguns dos parâmetros desta seção de um segredo dos AWS Secrets Manager (`sfUser`, `sfPassword`) e, nesse caso, não será necessário fornecê-los. O segredo deve fornecer o conteúdo protegido com as chaves `sfUser` e `sfPassword`. É possível fazer isso fornecendo o parâmetro `secretId`.

Os parâmetros a seguir geralmente são usados ao se conectar com o Snowflake.

- `sfDatabase`: obrigatório se o padrão do usuário não estiver definido no Snowflake. Usado para leitura/gravação. O banco de dados a ser usado na sessão após a conexão.

- `sfSchema`: obrigatório se o padrão do usuário não estiver definido no Snowflake. Usado para leitura/gravação. O esquema a ser usado na sessão após a conexão.
- `sfWarehouse`: obrigatório se o padrão do usuário não estiver definido no Snowflake. Usado para leitura/gravação. O armazém virtual padrão a ser usado na sessão após a conexão.
- `sfRole`: obrigatório se o padrão do usuário não estiver definido no Snowflake. Usado para leitura/gravação. O perfil de segurança padrão a ser usado na sessão após a conexão.
- `sfUrl`: (obrigatório) usado para leitura/gravação. Especifica o nome do host da sua conta no seguinte formato: *`account_identifier`*`.snowflakecomputing.com`. Para obter mais informações sobre identificadores de conta, consulte Account Identifiers na documentação do Snowflake.
- `sfUser`: (obrigatório) usado para leitura/gravação. Nome de login do usuário do Snowflake.
- `sfPassword`: (obrigatório, exceto se `pem_private_key` for fornecido) usado para leitura/gravação. Senha do usuário do Snowflake.
- `dbtable`: obrigatório ao trabalhar com tabelas completas. Usado para leitura/gravação. O nome da tabela a ser lida ou da tabela na qual os dados serão gravados. Durante a leitura, todas as colunas e registros são recuperados.
- `pem_private_key`: usado para leitura/gravação. Uma string de chave privada codificada em b64 não criptografada. A chave privada do usuário do Snowflake. É comum copiá-la de um arquivo PEM. Para obter mais informações, consulte Autenticação e rotação de pares de chaves na documentação do Snowflake.
- `query`: obrigatório ao ler com uma consulta. Usado para leitura. A consulta exata (instrução `SELECT`) a ser executada

As opções a seguir são usadas para configurar comportamentos específicos durante o processo de conexão com o Snowflake.

- `preactions`: usado para leitura/gravação. Valores válidos: lista separada por ponto e vírgula de instruções SQL como string. As instruções SQL são executadas antes que os dados sejam transferidos entre o AWS Glue e o Snowflake. Se uma instrução contiver `%s`, `%s` será substituído pelo nome da tabela referenciada para a operação.
- `postactions`: usado para leitura/gravação. As instruções SQL são executadas após os dados serem transferidos entre o AWS Glue e o Snowflake. Se uma instrução contiver `%s`, `%s` será substituído pelo nome da tabela referenciada para a operação.

- `autopushdown`: padrão: `"on"`. Valores válidos: `"on"`, `"off"`. Esse parâmetro controla se o pushdown automática de consultas está habilitado. Se o pushdown estiver habilitado, quando uma consulta for executada no Spark, se for possível fazer pushdown de parte da consulta para o servidor do Snowflake, isso ocorrerá. Isso melhora a performance de algumas consultas. Para obter informações sobre se é possível fazer pushdown da consulta, consulte Pushdown na documentação do Snowflake.

Além disso, algumas das opções disponíveis no conector do Snowflake Spark podem ser compatíveis com o AWS Glue. Para obter mais informações sobre as opções disponíveis no conector do Snowflake Spark, consulte Setting Configuration Options for the Connector no Snowflake.

Limitações do conector do Snowflake

A conexão com o Snowflake com o AWS Glue para Spark está sujeita às limitações a seguir.

- Esse conector não é compatível com marcadores de trabalho. Para obter mais informações sobre marcadores de trabalho, consulte the section called “Rastrear dados processados usando marcadores de trabalho”.
- Esse conector não é compatível com leituras e gravações do Snowflake por meio de tabelas no catálogo de dados do AWS Glue usando os métodos `create_dynamic_frame.from_catalog` e `write_dynamic_frame.from_catalog`.
- Esse conector não é compatível com a conexão com o Snowflake usando outras credenciais que não sejam usuário e senha.
- Esse conector não é compatível com trabalhos de streaming.
- Esse conector é compatível com consultas baseadas em instrução `SELECT` ao recuperar informações (como com o parâmetro `query`). Outros tipos de consultas (como `SHOW`, `DESC` ou instruções DML) não são compatíveis.
- O Snowflake limita o tamanho do texto da consulta (ou seja, instruções SQL) enviado pelos clientes do Snowflake a 1 MB por instrução. Para obter mais detalhes, consulte Limits on Query Text Size.

Conexões do Teradata Vantage

O AWS Glue para Spark pode ser usado para ler e escrever em tabelas no Teradata Vantage no AWS Glue 4.0 e versões posteriores. Você pode definir o que ler no Teradata com uma consulta

SQL. Conecte-se ao Teradata usando credenciais de nome de usuário e senha armazenadas no AWS Secrets Manager via conexão do AWS Glue.

Para obter mais informações sobre o Teradata, consulte a Documentação do Teradata.

Configurar conexões do Teradata

Para se conectar ao Teradata via AWS, será necessário criar e armazenar suas credenciais do Teradata em um segredo do AWS Secrets Manager e, em seguida, associar esse segredo a uma conexão ao AWS Glue do Teradata. Se sua instância do Teradata estiver em uma Amazon VPC, você também precisará fornecer opções de rede para sua conexão AWS Glue Teradata.

Para se conectar ao Teradata via AWS Glue, talvez seja necessário atender a alguns pré-requisitos:

- Se você estiver acessando seu ambiente Teradata via Amazon VPC, configure a Amazon VPC para permitir que seu trabalho do AWS Glue se comunique com o ambiente Teradata. Recomendamos não acessar o ambiente Teradata via Internet pública.

  Na Amazon VPC, identifique ou crie uma VPC, uma Sub-rede e um Grupo de segurança que o AWS Glue usará durante a execução do trabalho. Além disso, você precisa garantir que a Amazon VPC esteja configurada para permitir o tráfego de rede entre sua instância do Teradata e esse local. Seu trabalho precisará estabelecer uma conexão TCP com a porta cliente do Teradata. Para obter mais informações sobre portas do Teradata, consulte a Documentação do Teradata.

  Com base no layout da sua rede, a conectividade segura da VPC pode exigir alterações na Amazon VPC e em outros serviços de rede. Para obter mais informações sobre conectividade com a AWS, consulte Opções de conectividade da AWS na documentação da Teradata.

Para configurar uma conexão AWS Teradata:

1. Em sua configuração do Teradata, identifique ou crie um usuário e a senha com os quais o AWS Glue se conectará, *`teradataUser`* e *`teradataPassword`*. Para obter mais informações, consulte Visão geral da segurança do Vantage na documentação do Teradata.
2. No AWS Secrets Manager, crie um segredo usando suas credenciais do Teradata. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *`SecretName`*, para a próxima etapa.

- Ao selecionar Pares chave/valor, crie um par para a chave `user` com o valor *`teradataUsername`*.
- Ao selecionar Pares chave/valor, crie um par para a chave `password` com o valor *`teradataPassword`*.

3. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *`connectionName`*, para a próxima etapa.

   - Ao selecionar um Tipo de conexão, selecione Teradata.
   - Ao fornecer o URL do JDBC, forneça o URL da sua instância. Você também pode codificar certos parâmetros de conexão separados por vírgula no URL do JDBC. O URL deve estar de acordo com o seguinte formato: `jdbc:teradata://`*`teradataHostname`*`/`*`ParameterName`*`=`*`ParameterValue`*`,`*`ParameterName`*

     Os parâmetros de URL compatíveis incluem:

     - `DATABASE`: o nome do banco de dados no host a ser acessado por padrão.
     - `DBS_PORT`: a porta do banco de dados usada para execução em uma porta não padrão.
   - Ao selecionar um Tipo de credencial, selecione AWS Secrets Manager e defina Segredo da AWS como *`secretName`*.

4. Nas seguintes situações, configurações adicionais podem ser necessárias:

   - Para instâncias do Teradata hospedadas na AWS em uma Amazon VPC
     - Será necessário fornecer informações de conexão da Amazon VPC à conexão do AWS Glue que define suas credenciais de segurança do Teradata. Ao criar ou atualizar sua conexão, defina VPC, Sub-rede e Grupos de segurança em Opções de rede.

Depois de criar uma conexão AWS Glue Teradata, será necessário executar as etapas a seguir antes de chamar seu método de conexão.

- Conceda ao perfil do IAM associada ao seu trabalho do AWS Glue permissão para ler *`secretName`*.
- Na configuração do trabalho do AWS Glue, forneça *`connectionName`* como uma conexão de rede adicional.

Ler do Teradata

Pré-requisitos:

- Uma tabela do Teradata da qual você deseja ler. Você precisará do nome da tabela, *tableName*.
- Uma conexão AWS Glue Teradata configurada para fornecer informações de autenticação. Conclua as etapas Para configurar uma conexão com o Teradata para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *ConnectionName*.

Por exemplo:

```
teradata_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="teradata",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName"
    }
)
```

Você também pode fornecer uma consulta SQL SELECT para filtrar os resultados retornados ao seu DynamicFrame. Você precisará configurar `query`.

Por exemplo:

```
teradata_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="teradata",
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

Escrever em tabelas do Teradata

Pré-requisitos: uma tabela do Teradata na qual você gostaria de escrever, *tableName*. É necessário criar a tabela antes de chamar o método de conexão.

Por exemplo:

```
teradata_write = glueContext.write_dynamic_frame.from_options(
    connection_type="teradata",
```

```
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName"
    }
)
```

Referência de opções de conexão do Teradata

- `connectionName` — Obrigatório. Usado para leitura/gravação. O nome de uma conexão do AWS Glue Teradata configurada para fornecer informações de autenticação e rede ao seu método de conexão.
- `dbtable`: necessário para escrever, obrigatório para leitura, a menos que `query` seja fornecido. Usado para leitura/gravação. O nome de uma tabela com a qual seu método de conexão interagirá.
- `query` - Usado para leitura. Uma consulta SQL SELECT que define o que deve ser recuperado na leitura do Teradata.

Conexões do Vertica

O AWS Glue para Spark pode ser usado para ler e escrever em tabelas no Vertica no AWS Glue 4.0 e versões posteriores. Você pode definir o que ler no Vertica com uma consulta SQL. Conecte-se ao Vertica usando credenciais de nome de usuário e senha armazenadas no AWS Secrets Manager via conexão do AWS Glue.

Para obter mais informações sobre o Vertica, consulte a Documentação do Vertica.

Configurar conexões do Vertica

Para se conectar ao Vertica via AWS Glue, será necessário criar e armazenar suas credenciais do Vertica em um segredo do AWS Secrets Manager e, em seguida, associar esse segredo a uma conexão ao AWS Glue do Vertica. Se sua instância do Vertica estiver em uma Amazon VPC, você também precisará fornecer opções de rede para sua conexão AWS Glue Vertica. É necessário um bucket ou uma pasta do Amazon S3 para usar como armazenamento temporário ao ler e gravar no banco de dados.

Para se conectar ao Vertica via AWS Glue, talvez seja necessário atender a alguns pré-requisitos:

- Um bucket ou uma pasta do Amazon S3 para usar como armazenamento temporário ao ler e escrever no banco de dados, referido por *`tempS3Path`*.

> **Note**
>
> Quando o Vertica é usado em pré-visualizações de dados de trabalhos do AWS Glue, os arquivos temporários podem não ser removidos automaticamente de *tempS3Path*. Para garantir a remoção de arquivos temporários, encerre diretamente a sessão de visualização de dados escolhendo Encerrar sessão no painel Visualização de dados.
> Se não for possível garantir que a sessão de visualização de dados seja encerrada diretamente, considere definir a configuração do ciclo de vida do Amazon S3 para remover dados antigos. Recomendamos remover dados com mais de 49 horas com base no runtime máximo do trabalho somado a uma margem. Para obter mais informações sobre a configuração do Amazon S3, consulte Gerenciar o ciclo de vida do armazenamento na documentação do Amazon S3.

- Uma política do IAM com permissões apropriadas para seu caminho do Amazon S3 que você pode associar ao seu perfil de trabalho do AWS Glue.
- Se a sua instância do Vertica estiver em uma Amazon VPC, configure a Amazon VPC para permitir que seu trabalho do AWS Glue se comunique com a instância do Vertica sem que o tráfego passe pela Internet pública.

  Na Amazon VPC, identifique ou crie uma VPC, uma Sub-rede e um Grupo de segurança que o AWS Glue usará durante a execução do trabalho. Além disso, você precisa garantir que a Amazon VPC esteja configurada para permitir o tráfego de rede entre sua instância do Vertica e esse local. Seu trabalho precisará estabelecer uma conexão TCP com a porta cliente do Vertica (por padrão, 5433). Com base no layout da rede, isso pode exigir alterações em regras do grupo de segurança, ACLs de rede, gateways de NAT e conexões de emparelhamento.

Em seguida, você pode continuar com a configuração do AWS Glue para usá-lo com o Vertica.

Para configurar uma conexão com o Vertica:

1. No AWS Secrets Manager, crie um segredo usando suas credenciais do Vertica, *verticaUsername* e *verticaPassword*. Para criar um segredo no Secrets Manager, siga o tutorial disponível em Criar uma AWS Secrets Manager segredo na documentação do AWS Secrets Manager. Depois de criar o segredo, guarde o nome secreto, *SecretName*, para a próxima etapa.

- Ao selecionar Pares chave/valor, crie um par para a chave `user` com o valor *`verticaUsername`*.
- Ao selecionar Pares chave/valor, crie um par para a chave `password` com o valor *`verticaPassword`*.

2. No console do AWS Glue, crie uma conexão seguindo as etapas em the section called "Adicionar uma conexão do AWS Glue". Depois de criar a conexão, guarde o nome da conexão, *`connectionName`*, para a próxima etapa.

   - Ao selecionar um Tipo de conexão, selecione Vertica.
   - Ao selecionar Host do Vertica, forneça o URL da sua instalação do Vertica.
   - Ao selecionar Porta do Vertica, a porta pela qual sua instalação do Vertica está disponível.
   - Ao selecionar um Segredo da AWS, forneça o *`secretName`*.

3. Nas seguintes situações, configurações adicionais podem ser necessárias:

   - Para instâncias do Teradata hospedadas na AWS em uma Amazon VPC
     - Forneça informações de conexão da Amazon VPC à conexão do AWS Glue que define suas credenciais de segurança do Vertica. Ao criar ou atualizar sua conexão, defina VPC, Sub-rede e Grupos de segurança em Opções de rede.

Depois de criar uma conexão AWS Glue Vertica, será necessário executar as etapas a seguir antes de chamar seu método de conexão.

- Conceda ao perfil do IAM associado ao seu trabalho do AWS Glue permissão para *`tempS3Path`*.
- Conceda ao perfil do IAM associada ao seu trabalho do AWS Glue permissão para ler *`secretName`*.
- Na configuração do trabalho do AWS Glue, forneça *`connectionName`* como uma conexão de rede adicional.

Ler no Vertica

Pré-requisitos:

- Uma tabela do Vertica que você deseja ler. Você precisará do nome do banco de dados do Vertica, *`dbName`* e do nome da tabela, *`tableName`*.

- Uma conexão AWS Glue Vertica configurada para fornecer informações de autenticação. Conclua as etapas do procedimento anterior, Para configurar uma conexão com o Vertica para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *ConnectionName*.
- Um bucket ou pasta do Amazon S3 para usar para armazenamento temporário, mencionado anteriormente. Você precisará do nome *tempS3Path*. Você deverá se conectar a esse local usando o protocolo `s3a`.

Por exemplo:

```
dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="vertica",
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
        "db": "dbName",
        "table": "tableName",
    }
)
```

Você também pode fornecer uma consulta SQL SELECT para filtrar os resultados retornados ao seu DynamicFrame ou para acessar um conjunto de dados de várias tabelas.

Por exemplo:

```
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",
    },
)
```

Escrever em tabelas do Vertica

Este exemplo escreve informações de um DynamicFrame existente, *dynamicFrame*, no Vertica. Se a tabela já contiver informações, o AWS Glue anexará dados do seu DynamicFrame.

Pré-requisitos:

- O nome de uma tabela atual ou desejada, *tableName*, na qual você deseja escrever. Você também precisará do nome do banco de dados Vertica correspondente, *dbName*.
- Uma conexão AWS Glue Vertica configurada para fornecer informações de autenticação. Conclua as etapas do procedimento anterior, Para configurar uma conexão com o Vertica para configurar suas informações de autenticação. Você precisará do nome da conexão AWS Glue, *ConnectionName*.
- Um bucket ou pasta do Amazon S3 para usar para armazenamento temporário, mencionado anteriormente. Você precisará do nome *tempS3Path*. Você deverá se conectar a esse local usando o protocolo `s3a`.

Por exemplo:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="vertica",
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
        "db": "dbName",
        "table": "tableName",
    }
)
```

Referência de opções de conexão do Vertica

- `connectionName` — Obrigatório. Usado para leitura/gravação. O nome de uma conexão do AWS Glue Vertica configurada para fornecer informações de autenticação e rede ao seu método de conexão.
- db — Obrigatório. Usado para leitura/gravação. O nome de um banco de dados no Vertica com o qual seu método de conexão interagirá.
- `dbSchema`: exigido se necessário para identificar sua tabela. Usado para leitura/gravação. Padrão: `public`. O nome de um esquema com o qual seu método de conexão interagirá.
- `table`: necessário para escrever, obrigatório para leitura, a menos que `query` seja fornecido. Usado para leitura/gravação. O nome de uma tabela com a qual seu método de conexão interagirá.
- `query` - Usado para leitura. Uma consulta SQL SELECT que define o que deve ser recuperado na leitura do Teradata.

- `staging_fs_url` — Obrigatório. Usado para leitura/gravação. Valores válidos: URLs do `s3a`. O URL de um bucket ou pasta do Amazon S3 a ser usado para armazenamento temporário.

Valores de connectionType personalizados e AWS Marketplace

Incluindo o seguinte:

- `"connectionType": "marketplace.athena"`: designa uma conexão com um armazenamento de dados do Amazon Athena. A conexão usa um conector de AWS Marketplace.
- `"connectionType": "marketplace.spark"`: designa uma conexão com um armazenamento de dados do Apache Spark. A conexão usa um conector de AWS Marketplace.
- `"connectionType": "marketplace.jdbc"`: designa uma conexão com um armazenamento de dados do JDBC. A conexão usa um conector de AWS Marketplace.
- `"connectionType": "custom.athena"`: designa uma conexão com um armazenamento de dados do Amazon Athena. A conexão usa um conector personalizado que você carrega no AWS Glue Studio.
- `"connectionType": "custom.spark"`: designa uma conexão com um armazenamento de dados do Apache Spark. A conexão usa um conector personalizado que você carrega no AWS Glue Studio.
- `"connectionType": "custom.jdbc"`: designa uma conexão com um armazenamento de dados do JDBC. A conexão usa um conector personalizado que você carrega no AWS Glue Studio.

Opções de conexão para o tipo custom.jdbc ou marketplace.jdbc

- `className`: string, obrigatório, nome da classe do driver.
- `connectionName`: string, obrigatório, nome da conexão associada ao conector.
- `url`: string, obrigatório, URL do JDBC com espaços reservados (`${}`) que são usados para construir a conexão com a origem dos dados. O espaço reservado `${secretKey}` é substituído pelo segredo do mesmo nome em AWS Secrets Manager. Consulte a documentação do armazenamento de dados para obter mais informações sobre como construir o URL.
- `secretId` ou `user/password`: string, obrigatório, usado para recuperar credenciais para o URL.
- `dbTable` ou `query`: string, obrigatório, a tabela ou consulta SQL da qual obter os dados. Você pode especificar `dbTable` ou `query`, mas não os dois.

- `partitionColumn`: string, opcional, o nome de uma coluna de inteiros usada para o particionamento. Essa opção só funciona quando está incluída em `lowerBound`, `upperBound` e `numPartitions`. Essa opção funciona da mesma maneira que no leitor JDBC Spark SQL. Para obter mais informações, consulte JDBC para outros bancos de dados no Guia do Apache Spark SQL, DataFrames e conjuntos de dados.

  Os valores de `lowerBound` e `upperBound` são usados para decidir o passo de partição, não para filtrar as linhas na tabela. Todas as linhas na tabela são particionadas e retornadas.

  > **Note**
  >
  > Ao usar uma consulta em vez de um nome de tabela, você deve validar se a consulta funciona com a condição de particionamento especificada. Por exemplo:
  >
  > - Se o seu formato de consulta for `"SELECT col1 FROM table1"`, teste a consulta anexando uma cláusula `WHERE` no final da consulta que usa a coluna de partição.
  > - Se o seu formato de consulta for `"SELECT col1 FROM table1 WHERE col2=val"`, teste a consulta estendendo a cláusula `WHERE` com `AND` e uma expressão que usa a coluna de partição.

- `lowerBound`: inteiro, opcional, o valor mínimo de `partitionColumn` que é usado para decidir o passo de partição.
- `upperBound`: inteiro, opcional, o valor máximo de `partitionColumn` que é usado para decidir o passo de partição.
- `numPartitions`: inteiro, opcional, o número de partições. Esse valor, juntamente com `lowerBound` (inclusive) e `upperBound` (exclusive), forma os passos de partição para as expressões de cláusula `WHERE` geradas que são usadas para dividir a `partitionColumn`.

  > **Important**
  >
  > Tenha cuidado com a quantidade, pois muitas partições podem causar problemas em seus sistemas de banco de dados externo.

- `filterPredicate`: string, opcional, cláusula de condição extra para filtrar dados da fonte. Por exemplo:

```
BillingCity='Mountain View'
```

Ao usar uma consulta em vez de um nome de tabela, você deve validar que a consulta funciona com o `filterPredicate` especificado. Por exemplo:

- Se o seu formato de consulta for `"SELECT col1 FROM table1"`, teste a consulta anexando uma cláusula `WHERE` no final da consulta que usa o predicado do filtro.
- Se o seu formato de consulta for `"SELECT col1 FROM table1 WHERE col2=val"`, teste a consulta estendendo a cláusula `WHERE` com `AND` e uma expressão que usa o predicado do filtro.

- `dataTypeMapping`: dicionário, opcional, mapeamento de tipo de dados personalizado que constrói um mapeamento a partir de um tipo de dados JDBC para um tipo de dados Glue. Por exemplo, a opção `"dataTypeMapping":{"FLOAT":"STRING"}` mapeia campos de dados JDBC do tipo `FLOAT` para o tipo `String` do Java chamando o método `ResultSet.getString()` do driver e o usa para complilar registros do AWS Glue. O objeto `ResultSet` é implantado por cada driver, portanto, o comportamento é específico para o driver que você usa. Consulte a documentação do driver do JDBC para entender como ele executa as conversões.
- Os tipos de dados do AWS Glue compatíveis atualmente são:
  - DATA
  - STRING
  - TIMESTAMP
  - INT
  - FLOAT
  - LONG
  - BIGDECIMAL
  - BYTE
  - SHORT
  - DOUBLE

  Os tipos de dados JDBC compatíveis são Java8 java.sql.types.

  Os mapeamentos de tipos de dados padrão (de JDBC para o AWS Glue) são:

  - 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

Se você usar um mapeamento de tipo de dados personalizado com a opção `dataTypeMapping`, poderá substituir um mapeamento de tipo de dados padrão. Somente os tipos de dados JDBC listados na opção `dataTypeMapping` são afetados. O mapeamento padrão é usado para todos os outros tipos de dados JDBC. Você pode adicionar mapeamentos para tipos de dados JDBC adicionais, se necessário. Se um tipo de dados JDBC não estiver incluído no mapeamento padrão ou em um mapeamento personalizado, o tipo de dados será convertido para o tipo de dados `STRING` do AWS Glue por padrão.

Os exemplos de código Python a seguir mostram como fazer a leitura de bancos de dados JDBC com drivers JDBC AWS Marketplace. Ele demonstra a leitura de um banco de dados e a gravação em um local 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()
```

Opções de conexão para o tipo custom.athena ou marketplace.athena

- `className`: string, obrigatório, nome da classe do driver. Quando você estiver usando o conector Athena-CloudWatch, esse valor de parâmetro será o prefixo da classe Name (Nome) (por exemplo, `"com.amazonaws.athena.connectors"`). O conector Athena-CloudWatch é composto por duas classes: um manipulador de metadados e um manipulador de registros. Se você fornecer o prefixo comum aqui, a API carregará as classes corretas com base nesse prefixo.
- `tableName`: string, obrigatório, o nome do fluxo de log do CloudWatch a ser lido. Esse trecho de código usa o nome de exibição especial `all_log_streams`, o que significa que o quadro de dados dinâmicos retornada conterá dados de todos os fluxos de log no grupo de logs.
- `schemaName`: string, obrigatório, o nome do grupo de logs do CloudWatch a ser lido. Por exemplo, `./aws-glue/jobs/output`
- `connectionName`: string, obrigatório, nome da conexão associada ao conector.

Para obter opções adicionais para esse conector, consulte o arquivo Amazon Athena CloudWatch Connector README (LEIAME do conector Amazon Athena CloudWatch) no GitHub.

O exemplo de código Python a seguir mostra como fazer a leitura de um armazenamento de dados do Athena usando um conector AWS Marketplace. Ele demonstra a leitura do Athena e a gravação em um local 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()
```

Opções de conexão para o tipo custom.spark ou marketplace.spark

- `className`: string, obrigatório, nome da classe do conector.
- `secretId`: string, opcional, usado para recuperar credenciais para a conexão do conector.
- `connectionName`: string, obrigatório, nome da conexão associada ao conector.
- Outras opções dependem do armazenamento de dados. Por exemplo, as opções de configuração do OpenSearch começam com o prefixo `es`, conforme descrito na documentação Elasticsearch for Apache Hadoop (Elasticsearch para Apache Hadoop). Conexões do Spark com o Snowflake usam opções como `sfUser` e `sfPassword`, conforme descrito em Usar o conector do Spark no guia Conexão com o Snowflake.

O exemplo de código Python a seguir mostra como fazer a leitura de um armazenamento de dados do OpenSearch usando um conector `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()
```

Opções gerais

As opções nesta seção são fornecidas como `connection_options`, mas não se aplicam especificamente a um determinado conector.

Os parâmetros a seguir geralmente são usados ao configurar marcadores. Eles podem se aplicar aos fluxos de trabalho do Amazon S3 ou JDBC. Para ter mais informações, consulte the section called “Usar marcadores de trabalho”.

- `jobBookmarkKeys`: uma matriz de nomes de coluna.
- `jobBookmarkKeysSortOrder`: string que define como comparar valores com base na ordem de classificação. Valores válidos: `"asc"`, `"desc"`.
- `useS3ListImplementation`: usado para gerenciar a performance da memória ao listar o conteúdo dos buckets do Amazon S3. Para obter mais informações, consulte Optimize memory management in AWS Glue.

Opções de formato de dados para entradas e saídas no AWS Glue para Spark

Essas páginas oferecem informações sobre suporte a atributos e parâmetros de configuração para formatos de dados compatíveis com o AWS Glue para Spark. Consulte a seguir uma descrição do uso e da aplicabilidade destas informações.

Suporte de recursos em todos os formatos de dados no AWS Glue

Cada formato de dados pode ser compatível com diferentes recursos do AWS Glue. Os recursos comuns a seguir podem ou não ser compatíveis, de acordo com o tipo de formato. Consulte a documentação do formato de dados para entender como utilizar nossos recursos para atender a suas necessidades.

Leitura	O AWS Glue pode reconhecer e interpretar esse formato de dados sem recursos adicionais, como conectores.
Escrever	O AWS Glue pode gravar dados nesse formato sem recursos adicionais. É possível incluir bibliotecas de terceiros em seu trabalho e usar as funções padrão do Apache Spark para gravar dados, como faria em outros ambientes

	Spark. Para mais informações sobre como incluir bibliotecas, consulte the section called “Bibliotecas Python”.
Leitura de transmissão	O AWS Glue pode reconhecer e interpretar esse formato de dados a partir de um fluxo de mensagens do Apache Kafka, do Amazon Managed Streaming para Apache Kafka ou do Amazon Kinesis. Esperamos que os fluxos apresentem os dados em um formato consistente, para que sejam lidos como `DataFrames` .
Agrupar arquivos pequeno	O AWS Glue pode agrupar arquivos para trabalhos em lote enviados para cada nó durante a execução de transformações do AWS Glue. Isso pode melhorar consideravelmente a performance de workloads que envolvam grandes quantidades de arquivos pequenos. Para ter mais informações, consulte the section called “Agrupar arquivos de entrada”.
Marcadores de trabalho	O AWS Glue pode acompanhar o progresso das transformações que realizam o mesmo trabalho no mesmo conjunto de dados em todas as execuções de tarefas com indicadores de tarefas. Isso pode melhorar a performance de workloads que envolvem conjuntos de dados em que o trabalho só precisa ser feito com novos dados desde a última execução do trabalho. Para ter mais informações, consulte the section called “Rastrear dados processados usando marcadores de trabalho”.

Parâmetros usados para interagir com formatos de dados no AWSGlue

Certos tipos de conexão do AWS Glue são compatíveis com vários tipos de `format`, fazendo com que seja necessário especificar informações sobre o formato de dados com um objeto `format_options` ao usar métodos como `GlueContext.write_dynamic_frame.from_options`.

- `s3`: para obter mais informações, consulte Tipos de conexão e opções para ETL no AWS Glue: Parâmetros de conexão do S3. Também é possível visualizar a documentação para os métodos que facilitam esse tipo de conexão: the section called “create_dynamic_frame_from_options” e the section called “write_dynamic_frame_from_options” em Python e os métodos Scala correspondentes the section called “getSourceWithFormato” e the section called “getSinkWithFormato”.

- `kinesis`: para obter mais informações, consulte Tipos de conexão e opções para ETL no AWS Glue: Parâmetros de conexão do Kinesis. Também é possível visualizar a documentação para o método que facilita esse tipo de conexão: the section called “create_data_frame_from_options” em Python e o método Scala correspondente the section called “createDataFrameFromOptions”.
- `kafka`: para obter mais informações, consulte Tipos de conexão e opções para ETL no AWS Glue: Parâmetros de conexão do Kafka. Também é possível visualizar a documentação para o método que facilita esse tipo de conexão: the section called “create_data_frame_from_options” em Python e o método Scala correspondente the section called “createDataFrameFromOptions”.

Alguns tipos de conexão não exigem `format_options`. Por exemplo, em uso normal, uma conexão JDBC com um banco de dados relacional recupera dados em formato de dados tabular consistente. Portanto, a leitura de uma conexão JDBC não exigiria `format_options`.

Alguns métodos para ler e gravar dados no Glue não exigem `format_options`. Por exemplo, usar `GlueContext.create_dynamic_frame.from_catalog` com crawlers do AWS Glue. Os crawlers determinam a forma dos dados. Ao usar crawlers, um classificador do AWS Glue examinará seus dados para tomar decisões inteligentes sobre como representar o formato de dados. Em seguida, armazenará uma representação de seus dados no Catálogo de Dados do AWS Glue, que pode ser usado em um script ETL do AWS Glue para recuperar dados com o método `GlueContext.create_dynamic_frame.from_catalog`. Os crawlers eliminam a necessidade de especificar manualmente as informações sobre o formato dos dados.

Para trabalhos que acessam tabelas governadas pelo AWS Lake Formation, o AWS Glue oferece suporte para a leitura e a escrita de todos os formatos com suporte pelas tabelas governadas pelo

Lake Formation. Para obter a lista atual de formatos com suporte para tabelas governadas pelo AWS Lake Formation, consulte Notas e restrições para tabelas controladas no Guia do desenvolvedor do AWS Lake Formation.

> **Note**
>
> Para gravar Apache Parquet, o ETL do AWS Glue só oferece suporte para gravação em uma tabela controlada especificando uma opção para um tipo de gravador Parquet personalizado otimizado para quadros dinâmicos. Ao gravar em uma tabela governada com o formato `parquet`, você deve adicionar a chave `useGlueParquetWriter` com um valor de `true` nos parâmetros da tabela.

Tópicos

Uso do formato CSV no AWS Glue

O AWS Glue recupera dados de fontes e grava dados em destinos armazenados e transportados em vários formatos de dados. Se seus dados forem armazenados ou transportados no formato de dados CSV, este documento apresenta os recursos disponíveis para usar seus dados no AWS Glue.

O AWS Glue é compatível com o uso do formato de arquivo de valores separados por vírgulas (CSV). Esse formato é um formato de dados baseado em linha. Em geral, os CSVs não estão em conformidade estrita com um padrão, mas você pode consultar RFC 4180 e RFC 7111 para obter mais informações.

Você pode usar o AWS Glue para ler CSVs do Amazon S3 e de fontes de transmissão, bem como para gravar CSVs no Amazon S3. Você pode ler e gravar arquivos `bzip` e `gzip` do S3 que contenham arquivos CSV. Você configura o comportamento de compactação no Parâmetros de conexão do S3 e não na configuração apresentada nesta página.

A tabela a seguir mostra quais são os recursos comuns do AWS Glue compatíveis com a opção de formato CSV.

Leitura	Escrever	Leitura de transmissão	Agrupar arquivos pequenos	Marcadores de trabalho
Compatível	Compatível	Compatível	Compatível	Compatível

Exemplo: ler arquivos ou pastas CSV do S3

Pré-requisitos: você precisará dos caminhos do S3 (`s3path`) para os arquivos ou pastas CSV que deseja ler.

Configuração: nas opções da sua função, especifique `format="csv"`. Em seu `connection_options`, use a chave `paths` para especificar `s3path`. Você pode configurar como o leitor interage com o S3 no `connection_options`. Para mais detalhes, consulte os tipos de conexão e opções para ETL no AWS Glue: Parâmetros de conexão do S3. Você pode configurar como o leitor interpreta os arquivos CSV em seu `format_options`. Para obter mais detalhes, consulte CSV Configuration Reference (Referência de configuração de CSV).

O seguinte script de ETL do AWS Glue mostra o processo de leitura de arquivos ou pastas CSV provenientes do S3.

Fornecemos um leitor personalizado de CSV com otimizações de desempenho para fluxos de trabalho comuns por meio da chave de configuração `optimizePerformance`. Para determinar se esse leitor é adequado para sua workload, consulte the section called “Usar o leitor CSV otimizado”.

Python

Neste exemplo, use o 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,
    },
)
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .format("csv")\
    .option("header", "true")\
    .load("s3://s3path")
```

Scala

Neste exemplo, use a operação 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()
  }
}
```

Você também pode usar DataFrames em um script (`org.apache.spark.sql.DataFrame`).

```
val dataFrame = spark.read
  .option("header","true")
  .format("csv")
  .load("s3://s3path“)
```

Exemplo: gravar arquivos e pastas CSV no S3

Pré-requisitos: você precisará de um DataFrame (`dataFrame`) ou de um DynamicFrame (`dynamicFrame`) inicializado. Você também precisará do caminho de saída esperado do S3, `s3path`.

Configuração: nas opções da sua função, especifique `format="csv"`. Em seu `connection_options`, use a chave `paths` para especificar `s3path`. Você pode configurar como o gravador interage com o S3 no `connection_options`. Para mais detalhes, consulte os tipos de conexão e opções para ETL no AWS Glue: Parâmetros de conexão do S3. Você pode configurar como sua operação grava o conteúdo de seus arquivos no `format_options`. Para obter mais detalhes, consulte CSV Configuration Reference (Referência de configuração de CSV). O seguinte script de ETL do AWS Glue mostra o processo de gravação de arquivos e pastas CSV no S3.

Python

Neste exemplo, use o 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,
    },
)
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
dataFrame.write\
    .format("csv")\
    .option("quote", None)\
    .mode("append")\
    .save("s3://s3path")
```

Scala

Neste exemplo, use o 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)
  }
}
```

Você também pode usar DataFrames em um script (`org.apache.spark.sql.DataFrame`).

```
dataFrame.write
    .format("csv")
    .option("quote", null)
    .mode("Append")
    .save("s3://s3path")
```

Referência de configuração de CSV

Você pode usar o seguinte `format_options` sempre que as bibliotecas do AWS Glue especificarem `format="csv"`:

- `separator`: especifica o caractere delimitador. O padrão é uma vírgula, mas é possível especificar qualquer outro caractere.
  - Tipo: texto, padrão: `","`
- `escaper`: especifica um caractere a ser usado para escape. Essa opção é usada somente ao ler arquivos CSV, não para gravar. Se ativado, o caractere imediatamente seguinte será usado no estado em que se encontrar, exceto para um pequeno conjunto de escapes conhecidos (`\n`, `\r`, `\t` e `\0`).
  - Tipo: texto, padrão: nenhum
- `quoteChar`: especifica o caractere a ser usado para aspas. O padrão é aspas duplas. Defina como `-1` para desativar as aspas por completo.
  - Tipo: texto, padrão: `'"'`
- `multiLine`: especifica se um único registro pode abranger várias linhas. Isso pode ocorrer quando um campo contém um caractere de nova linha entre aspas. Você deve definir essa opção como `True` se algum registro ocupar várias linhas. Habilitar `multiLine` pode diminuir o desempenho, pois isso requer uma divisão de arquivos mais cautelosa durante a análise.
  - Tipo: booliano, padrão: `false`
- `withHeader`: especifica se a primeira linha deve ser tratada como um cabeçalho. Esta opção pode ser usada na classe `DynamicFrameReader`.

  - Tipo: booliano, padrão: `false`
- `writeHeader`: especifica se deve gravar o cabeçalho na saída. Esta opção pode ser usada na classe `DynamicFrameWriter`.
  - Tipo: booliano, padrão: `true`
- `skipFirst`: especifica se é necessário ignorar a primeira linha de dados.
  - Tipo: booliano, padrão: `false`
- `optimizePerformance`: especifica se deve usar o leitor de SIMD avançado para CSV junto com formatos de memória colunar baseados em Apache Arrow. Disponível apenas no AWS Glue 3.0+.
  - Tipo: booliano, padrão: `false`
- `strictCheckForQuoting`: ao escrever CSVs, o Glue pode adicionar aspas aos valores que interpreta como strings. Isso é feito para evitar ambiguidade no que é gravado. Para economizar tempo ao decidir o que gravar, o Glue pode incluir aspas em determinadas situações em que as aspas não são necessárias. Habilitar uma verificação rigorosa executará um cálculo mais intensivo e só incluirá aspas quando estritamente necessário. Disponível apenas no AWS Glue 3.0+.
  - Tipo: booliano, padrão: `false`

Otimizar o desempenho de leitura com o leitor de SIMD vetorizado para CSV

O AWS Glue versão 3.0 adiciona um leitor de CSV otimizado que pode acelerar significativamente o desempenho geral do trabalho em comparação com os leitores de CSV baseados em linhas.

O leitor otimizado:

- Usa instruções SIMD da CPU para fazer leituras do disco.
- Grava imediatamente registros na memória em um formato colunar (Apache Arrow).
- Divide os registros em lotes.

Isso economiza tempo de processamento quando os registros são colocados em lote ou convertidos em um formato colunar posteriormente. Por exemplo, ao alterar esquemas ou recuperar dados por coluna.

Para usar o leitor otimizado, defina `"optimizePerformance"` como `true` na propriedade `format_options` ou da tabela.

```
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")
```

Limitações para o leitor vetorizado de CSV

Observe as seguintes limitações do leitor vetorizado de CSV:

- Ele não oferece suporte às opções de formato `multiLine` e `escaper`. Ele usa o padrão `escaper` de caractere de aspas duplas `'"'`. Quando essas opções são definidas, o AWS Glue automaticamente volta ao uso do leitor de CSV baseado em linha.
- Ele não oferece suporte à criação de um DynamicFrame com ChoiceType.
- Ele não oferece suporte à criação de um DynamicFrame com registros de erro.
- Ele não oferece suporte à leitura de arquivos CSV com caracteres multibyte, como caracteres japoneses ou chineses.

Uso do formato Parquet no AWS Glue

O AWS Glue recupera dados de fontes e grava dados em destinos armazenados e transportados em vários formatos de dados. Se seus dados forem armazenados ou transportados no formato de dados Parquet, este documento apresenta os recursos disponíveis para usar seus dados no AWS Glue.

O AWS Glue é compatível com o uso do formato Parquet. Esse formato é um formato de dados orientado a desempenho e baseado em colunas. Para uma apresentação do formato feita pela autoridade padrão, consulte Apache Parquet Documentation Overview (Visão geral da documentação do Apache Parquet).

Você pode usar o AWS Glue para ler arquivos Parquet do Amazon S3 e de fontes de transmissão, bem como para gravar arquivos Parquet no Amazon S3. Você pode ler e gravar arquivos `bzip` e `gzip` do S3 que contenham arquivos Parquet. Você configura o comportamento de compactação no Parâmetros de conexão do S3 e não na configuração apresentada nesta página.

A tabela a seguir mostra quais são os recursos comuns do AWS Glue compatíveis com a opção de formato Parquet.

Leitura	Escrever	Leitura de transmissão	Agrupar arquivos pequenos	Marcadores de trabalho
Compatível	Compatível	Compatível	Sem suporte	Compatível*

*Compatível com o AWS Glue versão 1.0+

Exemplo: ler arquivos ou pastas Parquet do S3

Pré-requisitos: você precisará dos caminhos do S3 (`s3path`) para os arquivos ou pastas Parquet que deseja ler.

Configuração: nas opções da sua função, especifique `format="parquet"`. Em seu `connection_options`, use a chave `paths` para especificar seu `s3path`.

Você pode configurar como o leitor interage com o S3 no `connection_options`. Para mais detalhes, consulte os tipos de conexão e opções para ETL no AWS Glue: Parâmetros de conexão do S3.

Você pode configurar como o leitor interpreta os arquivos Parquet em seu `format_options`. Para obter mais detalhes, consulte Parquet Configuration Reference (Referência de configuração de Parquet).

O seguinte script de ETL do AWS Glue mostra o processo de leitura de arquivos ou pastas Parquet provenientes do S3:

Python

Neste exemplo, use o 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"
)
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read.parquet("s3://s3path/")
```

Scala

Neste exemplo, use o 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()
  }
}
```

Você também pode usar DataFrames em um script (`org.apache.spark.sql.DataFrame`).

```
spark.read.parquet("s3://s3path/")
```

Exemplo: gravar arquivos e pastas Parquet no S3

Pré-requisitos: você precisará de um DataFrame (`dataFrame`) ou de um DynamicFrame (`dynamicFrame`) inicializado. Você também precisará do caminho de saída esperado do S3, `s3path`.

Configuração: nas opções da sua função, especifique `format="parquet"`. Em seu `connection_options`, use a chave `paths` para especificar `s3path`.

Em `connection_options`, você pode alterar ainda mais a forma como o gravador interage com o S3. Para mais detalhes, consulte os tipos de conexão e opções para ETL no AWS Glue: Parâmetros de conexão do S3. Você pode configurar como sua operação grava o conteúdo de seus arquivos no `format_options`. Para obter mais detalhes, consulte Parquet Configuration Reference (Referência de configuração de Parquet).

O seguinte script de ETL do AWS Glue mostra o processo de gravação de arquivos e pastas Parquet no S3.

Fornecemos um gravador personalizado de Parquet com otimizações de desempenho para DynamicFrames, por meio da chave de configuração `useGlueParquetWriter`. Para determinar se esse gravador é adequado para sua workload, consulte Glue Parquet Writer (Gravador Parquet do Glue).

Python

Neste exemplo, use o 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,
    },
)
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
df.write.parquet("s3://s3path/")
```

Scala

Neste exemplo, use o 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)
  }
}
```

Você também pode usar DataFrames em um script (`org.apache.spark.sql.DataFrame`).

```
df.write.parquet("s3://s3path/")
```

Referência de configuração do Parquet

Você pode usar o seguinte `format_options` sempre que as bibliotecas do AWS Glue especificarem `format="parquet"`:

- `useGlueParquetWriter`: especifica o uso de um gravador de Parquet personalizado com otimizações de desempenho para fluxos de trabalho DynamicFrame. Para obter mais detalhes sobre o uso, consulte Glue Parquet Writer (Gravador Parquet do Glue).
  - Tipo: booliano, padrão:`false`

- `compression`: especifica o codec de compactação usado. Os valores são totalmente compatíveis com `org.apache.parquet.hadoop.metadata.CompressionCodecName`.
  - Tipo: texto enumerado, padrão: `"snappy"`
  - Valores: `"uncompressed"`, `"snappy"`, `"gzip"` e `"lzo"`
- `blockSize`: especifica em bytes o tamanho de um grupo de linhas que está sendo armazenado em memória. Você usa isso para ajustar o desempenho. O tamanho deve ser dividido exatamente em um número de megabytes.
  - Digite: numérico, padrão:`134217728`
  - O valor padrão é igual a 128 MB.
- `pageSize`: especifica o tamanho de uma página em bytes. Você usa isso para ajustar o desempenho. Uma página é a menor unidade que deve ser totalmente lida para acessar um único registro.
  - Digite: numérico, padrão:`1048576`
  - O valor padrão é igual a 1 MB.

> **Note**
>
> E ainda, quaisquer opções aceitas pelo código SparkSQL subjacente podem ser transmitidas para ele por meio do parâmetro de mapa `connection_options`. Por exemplo, é possível definir uma configuração do Spark, como mergeSchema, para o leitor de Spark do AWS Glue mesclar o esquema para todos os arquivos.

Otimizar o desempenho de gravação com o gravador de Parquet do AWS Glue

> **Note**
>
> Historicamente, o gravador de Parquet do AWS Glue tem sido acessado por meio do tipo de formato `glueparquet`. Esse padrão de acesso não é mais promovido. Em vez disso, use o tipo `parquet` com `useGlueParquetWriter` habilitado.

O gravador de Parquet do AWS Glue tem aprimoramentos de desempenho que permitem gravações mais rápidas de arquivos Parquet. O gravador tradicional calcula um esquema antes da gravação. O formato Parquet não armazena o esquema de uma maneira rapidamente recuperável, então

isso pode levar algum tempo. Com o gravador de Parquet do AWS Glue, não é necessário usar um esquema pré-calculado. O gravador calcula e modifica o esquema dinamicamente conforme recebe os dados.

Ao especificar `useGlueParquetWriter`, observe as seguintes limitações:

- O gravador oferece só é compatível com evolução do esquema (p. ex., adicionar ou remover colunas), mas não com a alteração de tipos de coluna, como com `ResolveChoice`.
- O gravador não oferece suporte à gravação de DataFrames vazios; por exemplo, para gravar um arquivo somente de esquema. Ao realizar a integração com o Catálogo de Dados da AWS ao configurar `enableUpdateCatalog=True`, a tentativa de gravar um DataFrame vazio não atualizará o Catálogo de Dados. Isso resultará na criação de uma tabela no Catálogo de Dados sem esquema.

Se sua transformação não exigir essas limitações, a ativação do gravador Parquet do AWS Glue deverá aumentar o desempenho.

Uso do formato XML no AWS Glue

O AWS Glue recupera dados de fontes e grava dados em destinos armazenados e transportados em vários formatos de dados. Se seus dados forem armazenados ou transportados no formato de dados XML, este documento apresenta os recursos disponíveis para usar seus dados no AWS Glue.

O AWS Glue é compatível com o uso do formato XML. Esse formato representa estruturas de dados altamente configuráveis e rigidamente definidas que não são baseadas em linhas ou colunas. O XML é altamente padronizado. Para uma introdução ao formato pela autoridade padrão, consulte XML Essentials (Conceitos básicos de XML).

Você pode usar o AWS Glue para ler arquivos XML do Amazon S3, bem como arquivos `bzip` e `gzip` contendo arquivos XML. Você configura o comportamento de compactação no Parâmetros de conexão do S3 e não na configuração apresentada nesta página.

A tabela a seguir mostra quais são os recursos comuns do AWS Glue compatíveis com a opção de formato XML.

Leitura	Escrever	Leitura de transmissão	Agrupar arquivos pequenos	Marcadores de trabalho
Compatível	Sem suporte	Sem suporte	Compatível	Compatível

Exemplo: ler XML do S3

O leitor de XML usa um nome de tag XML. Ele examina elementos com essa tag em sua entrada para inferir um esquema e preenche um DynamicFrame com valores correspondentes. A funcionalidade de XML do AWS Glue se comporta de modo semelhante a XML Data Source for Apache Spark (Fonte de dados XML para Apache Spark). Talvez você consiga obter informações sobre o comportamento básico comparando esse leitor com a documentação do projeto.

Pré-requisitos: você precisará dos caminhos do S3 (`s3path`) para os arquivos ou pastas XML que deseja ler, além de algumas informações sobre seu arquivo XML. Você também precisará da tag para o elemento XML que deseja ler, `xmlTag`.

Configuração: nas opções da sua função, especifique `format="xml"`. Em seu `connection_options`, use a chave `paths` para especificar `s3path`. Você pode configurar adicionalmente como o leitor interage com o S3 no `connection_options`. Para mais detalhes, consulte os tipos de conexão e opções para ETL no AWS Glue: Parâmetros de conexão do S3. Em seu `format_options`, use a chave `rowTag` para especificar `xmlTag`. Você pode configurar adicionalmente como o leitor interpreta os arquivos XML em seu `format_options`. Para obter mais detalhes, consulte XML Configuration Reference (Referência de configuração de XML).

O seguinte script de ETL do AWS Glue mostra o processo de leitura de arquivos ou pastas XML provenientes do S3.

Python

Neste exemplo, use o 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"},
```

```
)
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .format("xml")\
    .option("rowTag", "xmlTag")\
    .load("s3://s3path")
```

Scala

Neste exemplo, use a operação 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()
}
```

Você também pode usar DataFrames em um script (`org.apache.spark.sql.DataFrame`).

```
val dataFrame = spark.read
  .option("rowTag", "xmlTag")
  .format("xml")
  .load("s3://s3path“)
```

Referência de configuração de XML

Você pode usar o seguinte `format_options` sempre que as bibliotecas do AWS Glue especificarem `format="xml"`:

- `rowTag`: especifica a tag XML no arquivo a ser tratada como uma linha. Tags de linha não podem ser de fechamento automático.
  - Tipo: texto, obrigatório
- `encoding`: especifica a codificação de caracteres. Pode ser o nome ou alias de um Charset compatível com nosso ambiente de runtime. Não oferecemos garantias específicas em relação ao suporte à codificação, mas as codificações principais devem funcionar.
  - Tipo: texto, padrão: `"UTF-8"`
- `excludeAttribute`: especifica se você deseja excluir atributos em elementos ou não.
  - Tipo: booliano, padrão: `false`
- `treatEmptyValuesAsNulls`: especifica se o espaço em branco deve ser tratado como um valor nulo.
  - Tipo: booliano, padrão: `false`
- `attributePrefix`: um prefixo para atributos a fim de diferenciá-los de elementos de texto secundários. Esse prefixo é usado para nomes de campo.
  - Tipo: texto, padrão: `"_"`
- `valueTag`: a tag usada para um valor quando há atributos no elemento que não têm secundários.
  - Tipo: texto, padrão: `"_VALUE"`
- `ignoreSurroundingSpaces`: especifica se o espaço em branco que envolve valores deve ser ignorado.
  - Tipo: booliano, padrão: `false`
- `withSchema`: contém o esquema esperado, em situações nas quais você deseja substituir o esquema inferido. Se você não usar essa opção, o AWS Glue inferirá o esquema com base nos dados XML.
  - Tipo: texto, padrão: não aplicável
  - O valor deve ser um objeto JSON que represente um `StructType`.

Especificar manualmente o esquema XML

Exemplo de esquema XML manual

Este é um exemplo de uso da opção de formato `withSchema` para especificar o esquema para dados 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 do formato Avro no AWS Glue

O AWS Glue recupera dados de fontes e grava dados em destinos armazenados e transportados em vários formatos de dados. Se seus dados forem armazenados ou transportados no formato de dados Avro, o documento apresenta os recursos disponíveis para usar seus dados no AWS Glue.

O AWS Glue é compatível com o uso do formato Avro. Esse é um formato de dados orientado a performance e baseado em linhas. Para uma apresentação do formato feita pela autoridade padrão, consulte a documentação do Apache Avro 1.8.2.

Você pode usar o AWS Glue para ler arquivos Avro do Amazon S3 e de fontes de transmissão, bem como para gravar arquivos Avro no Amazon S3. Você pode ler e gravar arquivos `bzip2` e `gzip` do S3 que contenham arquivos Avro. Além disso, você pode gravar arquivamentos `deflate`, `snappy` e `xz` contendo arquivos Avro. Você configura o comportamento de compactação no Parâmetros de conexão do S3 e não na configuração apresentada nesta página.

A tabela a seguir mostra quais são as operações comuns do AWS Glue compatíveis com a opção de formato Avro.

Leitura	Escrever	Leitura de transmissão	Agrupar arquivos pequenos	Marcadores de trabalho
Compatível	Compatível	Compatível*	Sem suporte	Compatível

*Compatível, com restrições. Para ter mais informações, consulte the section called “Notas e restrições para fontes de transmissão Avro”.

Exemplo: ler arquivos ou pastas Avro do S3

Pré-requisitos: você precisará dos caminhos do S3 (`s3path`) para os arquivos ou pastas Avro que deseja ler.

Configuração: nas opções da sua função, especifique `format="avro"`. Em seu `connection_options`, use a chave `paths` para especificar `s3path`. Você pode configurar como o leitor interage com o S3 no `connection_options`. Para obter detalhes, consulte Opções de formato de dados para entradas e saídas de ETL no AWSGlue: the section called “Parâmetros de conexão do S3”. Você pode configurar como o leitor interpreta os arquivos Avro em seu `format_options`. Para obter mais detalhes, consulte Avro Configuration Reference (Referência de configuração de Avro).

O seguinte script de ETL do AWS Glue mostra o processo de leitura de arquivos ou pastas Avro provenientes do S3:

Python

Neste exemplo, use o 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

Neste exemplo, use a operação 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()
  }
```

Exemplo: gravar arquivos e pastas Avro no S3

Pré-requisitos: você precisará de um DataFrame (`dataFrame`) ou de um DynamicFrame (`dynamicFrame`) inicializado. Você também precisará do caminho de saída esperado do S3, `s3path`.

Configuração: nas opções da sua função, especifique `format="avro"`. Em seu `connection_options`, use a chave `paths` para especificar seu `s3path`. Em `connection_options`, você pode alterar ainda mais a forma como o gravador interage com o S3. Para obter detalhes, consulte Opções de formato de dados para entradas e saídas de ETL no AWSGlue: the section called “Parâmetros de conexão do S3”. Você pode alterar a forma como o gravador interpreta os arquivos Avro em `format_options`. Para obter mais detalhes, consulte Avro Configuration Reference (Referência de configuração de Avro).

O seguinte script de ETL do AWS Glue mostra o processo de gravação de arquivos ou pastas Avro no S3.

Python

Neste exemplo, use o 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

Neste exemplo, use o 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)
  }
}
```

Referência de configuração de Avro

Você pode usar os seguinte valores de `format_options` sempre que as bibliotecas do AWS Glue especificarem `format="avro"`:

- `version`: especifica a versão do formato de leitor/gravador Apache Avro a ser suportada. O padrão é “1.7”. É possível especificar `format_options={"version": “1.8”}` para habilitar a leitura e gravação do tipo lógico Avro. Para obter mais informações, consulte a Especificação do Apache Avro 1.7.7 e a Especificação do Apache Avro 1.8.2.

  O conector do Apache Avro 1.8 oferece suporte para os seguintes tipos de conversão:

Para o leitor: esta tabela mostra a conversão entre o tipo de dados Avro (tipo lógico e o tipo primitivo Avro) e o tipo de dados `DynamicFrame` do AWS Glue para o leitor Avro 1.7 e 1.8.

Tipo de dados Avro: Tipo lógico	Tipo de dados Avro: Tipo primitivo Avro	Tipo de dados GlueDynamicFrame: Leitor Avro 1.7	Tipo de dados GlueDynamicFrame: Leitor Avro 1.8
Decimal	bytes	BINARY	Decimal
Decimal	fixo	BINARY	Decimal
Data	int	INT	Data
Tempo (milissegundo)	int	INT	INT
Tempo (microsse gundo)	longo	LONG	LONG
Time stamp (milisseg undo)	longo	LONG	Timestamp
Time stamp (microsse gundo)	longo	LONG	LONG
Duração (não é um tipo lógico)	fixo de 12	BINARY	BINARY

Para o gravador: esta tabela mostra a conversão entre o tipo de dados `DynamicFrame` do AWS Glue e o tipo de dados Avro para o gravador Avro 1.7 e 1.8.

Tipo de dados do **`DynamicFrame`** do AWS Glue	Tipo de dados Avro: Gravador Avro 1.7	Tipo de dados Avro: Gravador Avro 1.8
Decimal	String	decimal
Data	String	data
Marca de data e hora	String	timestamp-micros

Suporte para Avro Spark DataFrame

Para usar o Avro da API Spark DataFrame, é necessário instalar o plugin Spark Avro para a versão correspondente do Spark. A versão do Spark disponível em seu trabalho é determinada pela versão do AWS Glue. Para obter mais informações sobre as versões do Spark, consulte the section called "Versões do AWS Glue". O plugin é mantido pelo Apache. Não oferecemos garantias específicas de suporte.

No AWS Glue 2.0: use a versão 2.4.3 do plugin do Spark Avro. Você encontra esse JAR no Maven Central. Consulte org.apache.spark:spark-avro_2.12:2.4.3.

No AWS Glue 3.0: use a versão 3.1.1 do plugin do Spark Avro. Você encontra esse JAR no Maven Central. Consulte org.apache.spark:spark-avro_2.12:3.1.1.

Para incluir JARs adicionais em um trabalho de ETL do AWS Glue, use o parâmetro de trabalho `--extra-jars`. Para obter mais informações sobre parâmetros de trabalho, consulte the section called "Parâmetros de trabalho". Também é possível configurar esse parâmetro no AWS Management Console.

Uso do formato grokLog no AWS Glue

O AWS Glue recupera dados de fontes e grava dados em destinos armazenados e transportados em vários formatos de dados. Se seus dados forem armazenados ou transportados em um formato de texto não criptografado fracamente estruturado, o documento apresenta os recursos disponíveis para usar seus dados no AWS Glue por meio de padrões Grok.

O AWS Glue é compatível com o uso de padrões Grok. Os padrões Grok são semelhantes aos grupos de captura de expressões regulares. Eles reconhecem padrões de sequências de caracteres em um arquivo de texto simples e fornecem um tipo e propósito. No AWS Glue, seu objetivo principal

é ler logs. Para ver uma apresentação do Grok feita pelos autores, consulte Logstash Reference: Grok filter plugin (Referência do Logstash: plugin de filtro Grok).

Leitura	Escrever	Leitura de transmissão	Agrupar arquivos pequenos	Marcadores de trabalho
Compatível	Não aplicável	Compatível	Compatível	Sem suporte

Referência da configuração grokLog

Você pode usar os seguintes valores de `format_options` com `format="grokLog"`:

- `logFormat`: especifica o padrão Grok que corresponde ao formato de log.
- `customPatterns`: especifica outros padrões Grok usados aqui.
- `MISSING`: especifica o sinal a ser usado na identificação de valores ausentes. O padrão é `'-'`.
- `LineCount`: especifica o número de linhas em cada registro de log. O padrão é `'1'`, e atualmente somente os registros de linha única são compatíveis.
- `StrictMode`: um valor booliano que especifica se o modo estrito está habilitado. No modo estrito, o leitor não faz conversão ou recuperação de tipo automática. O valor padrão é `"false"`.

Uso do formato Ion no AWS Glue

O AWS Glue recupera dados de fontes e grava dados em destinos armazenados e transportados em vários formatos de dados. Se seus dados forem armazenados ou transportados no formato de dados Ion, este documento apresenta os recursos disponíveis para usar seus dados no AWS Glue.

O AWS Glue é compatível com o uso do formato Ion. Esse formato representa estruturas de dados (que não são baseadas em linhas ou colunas) em representações intercambiáveis em binários e em texto não criptografado. Para ver uma apresentação do formato feita pelos autores, consulte Amazon Ion. (Para obter mais informações, consulte a Especificação do Amazon Ion.)

Você pode usar o AWS Glue para ler arquivos Ion do Amazon S3. Você pode ler arquivos `bzip` e `gzip` do S3 que contenham arquivos Ion. Você configura o comportamento de compactação no Parâmetros de conexão do S3 e não na configuração apresentada nesta página.

A tabela a seguir mostra quais são as operações comuns do AWS Glue compatíveis com a opção de formato Ion.

Leitura	Escrever	Leitura de transmissão	Agrupar arquivos pequenos	Marcadores de trabalho
Compatível	Sem suporte	Sem suporte	Compatível	Sem suporte

Exemplo: ler arquivos e pastas Ion do S3

Pré-requisitos: você precisará dos caminhos do S3 (`s3path`) para os arquivos ou pastas Ion que você deseja ler.

Configuração: nas opções da sua função, especifique `format="json"`. Em seu `connection_options`, use a chave `paths` para especificar seu `s3path`. Você pode configurar como o leitor interage com o S3 no `connection_options`. Para mais detalhes, consulte os tipos de conexão e opções para ETL no AWS Glue: the section called “Parâmetros de conexão do S3”.

O seguinte script de ETL do AWS Glue mostra o processo de leitura de arquivos ou pastas Ion provenientes do S3:

Python

Neste exemplo, use o 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

Neste exemplo, use a operação 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()
  }
}
```

Referência de configuração de Ion

Não há valores de `format_options` para `format="ion"`.

Usando o formato JSON no AWS Glue

AWS O Glue recupera dados de fontes e grava dados em destinos armazenados e transportados em vários formatos de dados. Se seus dados forem armazenados ou transportados no formato de dados JSON, este documento apresenta os recursos disponíveis para usar seus dados no Glue. AWS

AWS O Glue suporta o uso do formato JSON. Esse formato representa estruturas de dados com forma consistente, mas com conteúdo flexível, que não são baseadas em linhas ou colunas. O JSON é definido por padrões paralelos emitidos por várias autoridades, e uma dela é a ECMA-404. Para ver uma apresentação do formato por uma fonte bastante referenciada, consulte Introducing JSON (Introdução ao JSON).

Você pode usar o AWS Glue para ler arquivos JSON do Amazon S3, bem `bzip` como arquivos JSON `gzip` compactados. Você configura o comportamento de compactação no Parâmetros de conexão do S3 e não na configuração apresentada nesta página.

Leitura	Escrever	Leitura de transmissão	Agrupar arquivos pequenos	Marcadores de trabalho	
Compatível	Compatível	Compatível	Compatível	Compatível	

Exemplo: ler arquivos ou pastas JSON do S3

Pré-requisitos: você precisará dos caminhos do S3 (`s3path`) para os arquivos ou pastas JSON que gostaria de ler.

Configuração: nas opções da sua função, especifique `format="json"`. Em seu `connection_options`, use a chave `paths` para especificar seu `s3path`. Você pode alterar ainda mais a forma como sua operação de leitura atravessará o S3 nas opções de conexão. Consulte the section called “Parâmetros de conexão do S3” para obter mais detalhes. Você pode configurar como o leitor interpreta os arquivos JSON em seu `format_options`. Para obter mais detalhes, consulte JSON Configuration Reference (Referência de configuração de JSON).

O script AWS Glue ETL a seguir mostra o processo de leitura de arquivos ou pastas JSON do S3:

Python

Neste exemplo, use o 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
    }
)
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .option("multiLine", "true")\
    .json("s3://s3path")
```

Scala

Para este exemplo, use a operação getSourceWithFormatar.

```
// 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()
  }
}
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
val dataFrame = spark.read
    .option("multiLine", "true")
    .json("s3://s3path")
```

Exemplo: gravar arquivos e pastas JSON no S3

Pré-requisitos: Você precisará de um inicializado DataFrame (`dataFrame`) ou (). DynamicFrame `dynamicFrame` Você também precisará do caminho de saída esperado do S3, `s3path`.

Configuração: nas opções da sua função, especifique `format="json"`. Em seu `connection_options`, use a chave `paths` para especificar `s3path`. Em `connection_options`, você pode alterar ainda mais a forma como o gravador interage com o S3. Para obter detalhes, consulte Opções de formato de dados para entradas e saídas ETL no AWS Glue:. the section called "Parâmetros de conexão do S3" Você pode configurar como o gravador interpreta os arquivos JSON em seu `format_options`. Para obter mais detalhes, consulte JSON Configuration Reference (Referência de configuração de JSON).

O script AWS Glue ETL a seguir mostra o processo de gravação de arquivos ou pastas JSON a partir do S3:

Python

Neste exemplo, use o 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"
)
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
df.write.json("s3://s3path/")
```

Scala

Neste exemplo, use o 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)
  }
}
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
df.write.json("s3://s3path")
```

Referência de configuração de JSON

Você pode usar os seguintes valores de `format_options` com `format="json"`:

- `jsonPath`— Uma JsonPathexpressão que identifica um objeto a ser lido em registros. Isso é especialmente útil quando um arquivo contém registros aninhados dentro de uma matriz externa. Por exemplo, a JsonPath expressão a seguir tem como alvo o `id` campo de um objeto JSON.

```
format="json", format_options={"jsonPath": "$.id"}
```

- `multiLine`: um valor booliano que especifica se um único registro pode gerar várias linhas. Isso pode ocorrer quando um campo contém um caractere de nova linha entre aspas. Você deve definir

essa opção como `"true"` se algum registro ocupar várias linhas. O valor padrão é `"false"`, que permite uma divisão de arquivos mais radical durante a análise.

- `optimizePerformance`: um valor booleano que especifica se deve usar o leitor SIMD para JSON avançado junto com formatos de memória colunar baseados em Apache Arrow. Disponível apenas no AWS Glue 3.0. Não compatível com `multiLine` ou `jsonPath`. Fornecer qualquer uma dessas opções instruirá o AWS Glue a usar o leitor padrão.
- `withSchema`: um valor de String que especifica um esquema de tabela no formato descrito em the section called “Especificar esquema XML”. Usado apenas com o `optimizePerformance` ao ler de conexões que não são do catálogo.

Usar o leitor vetorizado SIMD para JSON com formato colunar Apache Arrow

O AWS Glue versão 3.0 adiciona um leitor vetorizado para dados JSON. Ele executa duas vezes mais rápido sob certas condições, em comparação com o leitor padrão. Esse leitor vem com certas limitações que os usuários devem conhecer antes de usar, documentadas nesta seção.

Para usar o leitor otimizado, defina `"optimizePerformance"` como Verdadeiro no `format_options` ou propriedade de tabela. Você também precisará fornecer `withSchema` a menos que esteja lendo do catálogo. O `withSchema` espera uma entrada conforme descrito em the section called “Especificar 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 obter mais informações sobre o edifício a *SchemaString*na biblioteca AWS Glue, consultethe section called “Tipos”.

Limitações para o leitor vetorizado de CSV

Observe as seguintes limitações:

- Elementos JSON com objetos aninhados ou valores de matriz não são compatíveis. Se fornecido, o AWS Glue retornará ao leitor padrão.
- Um esquema deve ser fornecido do Catálogo ou com `withSchema`.
- Não compatível com `multiLine` ou `jsonPath`. Fornecer qualquer uma dessas opções instruirá o AWS Glue a usar o leitor padrão.
- Fornecer registros de entrada que não correspondam ao esquema de entrada resultará em falha do leitor.
- Registros de erro não serão criados.
- Arquivos JSON com caracteres multiByte, como caracteres japoneses ou chineses, não são aceitos.

Usar o formato ORC no AWS Glue

O AWS Glue recupera dados de fontes e grava dados em destinos armazenados e transportados em vários formatos de dados. Se seus dados forem armazenados ou transportados no formato de dados ORC, este documento apresenta os recursos disponíveis para usar seus dados no AWS Glue.

O AWS Glue é compatível com o uso do formato ORC. Esse formato é um formato de dados orientado a desempenho e baseado em colunas. Para ver uma introdução ao formato pela autoridade padrão, consulte Apache Orc.

Você pode usar o AWS Glue para ler arquivos ORC do Amazon S3 e de fontes de transmissão, bem como para gravar arquivos ORC no Amazon S3. Você pode ler e gravar arquivos `bzip` e `gzip` do S3 que contenham arquivos ORC. Você configura o comportamento de compactação no Parâmetros de conexão do S3 e não na configuração apresentada nesta página.

A tabela a seguir mostra quais são as operações comuns do AWS Glue compatíveis com a opção de formato ORC.

Leitura	Escrever	Leitura de transmissão	Agrupar arquivos pequenos	Marcadores de trabalho
Compatível	Compatível	Compatível	Sem suporte	Compatível*

*Compatível com o AWS Glue versão 1.0+

Exemplo: ler arquivos ou pastas ORC do S3

Pré-requisitos: você precisará dos caminhos do S3 (`s3path`) para os arquivos ou pastas ORC que deseja ler.

Configuração: nas opções da sua função, especifique `format="orc"`. Em seu `connection_options`, use a chave `paths` para especificar seu `s3path`. Você pode configurar como o leitor interage com o S3 no `connection_options`. Para mais detalhes, consulte os tipos de conexão e opções para ETL no AWS Glue: the section called “Parâmetros de conexão do S3”.

O seguinte script de ETL do AWS Glue mostra o processo de leitura de arquivos ou pastas ORC provenientes do S3:

Python

Neste exemplo, use o 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"
)
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .orc("s3://s3path")
```

Scala

Neste exemplo, use a operação 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()
  }
}
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
val dataFrame = spark.read
    .orc("s3://s3path")
```

Exemplo: gravar arquivos e pastas ORC no S3

Pré-requisitos: você precisará de um DataFrame (`dataFrame`) ou de um DynamicFrame (`dynamicFrame`) inicializado. Você também precisará do caminho de saída esperado do S3, `s3path`.

Configuração: nas opções da sua função, especifique `format="orc"`. Em suas opções de conexão, use a chave `paths` para especificar `s3path`. Em `connection_options`, você pode alterar ainda mais a forma como o gravador interage com o S3. Para obter detalhes, consulte Opções de formato de dados para entradas e saídas de ETL no AWSGlue: the section called “Parâmetros de conexão do S3”. O código de exemplo a seguir mostra o processo:

Python

Neste exemplo, use o 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"
    }
)
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
df.write.orc("s3://s3path/")
```

Scala

Neste exemplo, use o 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)
  }
}
```

Você também pode usar DataFrames em um script (`pyspark.sql.DataFrame`).

```
df.write.orc("s3://s3path/")
```

Referência de configuração de ORC

Não há valores de `format_options` para `format="orc"`. No entanto, quaisquer opções aceitas pelo código SparkSQL subjacente podem ser transmitidas para ele por meio do parâmetro de mapa `connection_options`.

Usar estruturas de data lake com trabalhos do AWS Glue ETL

As estruturas de data lake de código aberto simplificam o processamento incremental de dados para os arquivos que você armazena em data lakes criados no Amazon S3. O AWS Glue 3.0 e posteriores são compatíveis com as seguintes estruturas de data lake de código aberto:

- Apache Hudi
- Linux Foundation Delta Lake
- Apache Iceberg

Fornecemos suporte nativo para essas estruturas para que você possa ler e gravar os dados que armazenar no Amazon S3 de maneira transacionalmente consistente. Não é necessário instalar um conector separado nem realizar etapas adicionais de configuração para usar essas estruturas em trabalhos do AWS Glue ETL.

Ao gerenciar conjuntos de dados por meio do AWS Glue Data Catalog, você pode usar os métodos do AWS Glue para ler e gravar tabelas de data lake com o Spark DataFrames. Você também pode ler e gravar dados do Amazon S3 usando a API DataFrame do Spark.

Neste vídeo, você pode aprender sobre os conceitos básicos de como o Apache Hudi, o Apache Iceberg e o Delta Lake funcionam. Você verá como inserir, atualizar e excluir dados no data lake e como cada uma dessas estruturas funciona.

Tópicos

Limitações

Considere as seguintes limitações antes de usar estruturas de data lake com o. AWS Glue

- Os AWS Glue `GlueContext` métodos a seguir DynamicFrame não oferecem suporte à leitura e gravação de tabelas da estrutura do data lake. Em vez disso, use os `GlueContext` métodos for DataFrame ou a DataFrame API Spark.
  - Os `GlueContext` métodos a seguir não DynamicFrame são compatíveis com o controle de permissão do Lake Formation:
    - `create_dynamic_frame.from_catalog`
    - `write_dynamic_frame.from_catalog`
    - `getDynamicFrame`
    - `writeDynamicFrame`
  - Os `GlueContext` métodos a seguir DataFrame são compatíveis com o controle de permissão do Lake Formation:
    - `create_data_frame.from_catalog`
    - `write_data_frame.from_catalog`
    - `getDataFrame`
    - `writeDataFrame`
- Agrupamento de arquivos pequenos não é compatível.
- Marcadores de trabalho não são compatíveis.
- O Apache Hudi 0.10.1 para AWS Glue 3.0 não suporta tabelas Hudi Merge on Read (MoR).
- `ALTER TABLE … RENAME TO`não está disponível para o Apache Iceberg 0.13.1 for 3.0. AWS Glue

Limitações para tabelas em formato de data lake gerenciadas pelas permissões do Lake Formation

Os formatos de data lake são integrados ao AWS Glue ETL por meio das permissões do Lake Formation. A criação de um DynamicFrame uso não `create_dynamic_frame` é suportada. Para obter mais informações, veja os exemplos a seguir:

- Exemplo: ler e escrever na tabela do Iceberg com o controle de permissão do Lake Formation
- Exemplo: ler e escrever na tabela do Hudi com o controle de permissão do Lake Formation
- Exemplo: ler e escrever na tabela do Delta Lake com o controle de permissão do Lake Formation

> **Note**
>
> A integração com AWS Glue ETL por meio das permissões do Lake Formation para Apache Hudi, Apache Iceberg e Delta Lake é suportada somente na versão 4.0. AWS Glue

O Apache Iceberg tem a melhor integração com o AWS Glue ETL por meio das permissões do Lake Formation. Ele é compatível com quase todas as operações e inclui suporte a SQL.

O Hudi é compatível com a maioria das operações básicas, com exceção de operações administrativas. Isso ocorre porque essas opções geralmente são feitas por meio da gravação de dataframes e especificadas via `additional_options`. Você precisa usar AWS Glue APIs DataFrames para criar suas operações, pois o SparkSQL não é suportado.

O Delta Lake é compatível somente com leitura, anexação e substituição de dados da tabela. O Delta Lake exige o uso de suas próprias bibliotecas para poder realizar várias tarefas, como atualizações.

Os recursos a seguir não estão disponíveis para tabelas do Iceberg gerenciadas por permissões do Lake Formation.

- Compactação usando ETL AWS Glue
- Suporte ao Spark SQL via AWS Glue ETL

A seguir estão as limitações das tabelas do Hudi gerenciadas por permissões do Lake Formation:

- Remoção de arquivos órfãos

A seguir estão as limitações das tabelas do Data Lake gerenciadas por permissões do Lake Formation:

- Todos os recursos, exceto inserir e ler das tabelas do Delta Lake.

Usar a estrutura Hudi no AWS Glue

O AWS Glue 3.0 e versões posteriores são compatíveis com a estrutura Apache Hudi para data lakes. Hudi é uma estrutura de armazenamento de data lake de código aberto que simplifica o processamento incremental de dados e o desenvolvimento de pipelines de dados. Este tópico

aborda os recursos disponíveis para usar dados no AWS Glue ao transportar ou armazenar dados em uma tabela do Hudi. Para saber mais sobre o Hudi, consulte a documentação do Apache Hudi.

Você pode usar o AWS Glue para realizar operações de leitura e gravação em tabelas do Hudi no Amazon S3 ou trabalhar com tabelas do Hudi usando o AWS Glue Data Catalog. Operações adicionais, incluindo inserção, atualização e todas as operações do Apache Spark também são compatíveis.

> **Note**
>
> O Apache Hudi 0.10.1 para AWS Glue 3.0 não é compatível com tabelas Merge on Read (MoR) do Hudi.

A tabela a seguir lista a versão do Hudi incluída em cada versão do AWS Glue.

Versão do AWS Glue	Versões compatíveis do Hudi
4,0	0.12.1
3.0	0.10.1

Para saber mais sobre as estruturas de data lake compatíveis com o AWS Glue, consulte Usar estruturas de data lake com trabalhos do AWS Glue ETL.

Habilitar o Hudi

Para habilitar o Hudi para AWS Glue, faça as seguintes tarefas:

- Especifique `hudi` como um valor para o parâmetro de trabalho `--datalake-formats`. Para ter mais informações, consulte Parâmetros de trabalho do AWS Glue.
- Crie uma chave denominada `--conf` para o trabalho do AWS Glue e defina-a com o seguinte valor. Ou então, você pode definir a configuração a seguir usando `SparkConf` no script. Essas configurações ajudam o Apache Spark a lidar corretamente com as tabelas do Hudi.

```
spark.serializer=org.apache.spark.serializer.KryoSerializer --conf
 spark.sql.hive.convertMetastoreParquet=false
```

- O suporte à permissão do Lake Formation para tabelas do Hudi está habilitado por padrão para o AWS Glue 4.0. Nenhuma configuração adicional é necessária para leitura/escrita em tabelas do Hudi registradas no Lake Formation. Para ler uma tabela do Hudi registrada, o perfil do IAM do trabalho do AWS Glue deve ter a permissão SELECT. Para escrever em uma tabela do Hudi registrada, o perfil do IAM do trabalho do AWS Glue deve ter a permissão SUPER. Para saber mais sobre como gerenciar as permissões do Lake Formation, consulte Conceder e revogar permissões nos recursos do Catálogo de Dados.

Usar uma versão diferente do Hudi

Para usar uma versão do Hudi que não é compatível com o AWS Glue, especifique seus próprios arquivos JAR do Hudi usando o parâmetro de trabalho `--extra-jars`. Não inclua `hudi` como um valor para o parâmetro de trabalho `--datalake-formats`.

Exemplo: escrever uma tabela Hudi no Amazon S3 e registrá-la no AWS Glue Data Catalog

O exemplo a seguir demonstra como gravar uma tabela do Hudi no Amazon S3 e registrá-la no AWS Glue Data Catalog. O exemplo usa a ferramenta Hive Sync do Hudi para registrar a tabela.

> **Note**
>
> Este exemplo exige que você defina o parâmetro de trabalho `--enable-glue-datacatalog` para usar o AWS Glue Data Catalog como metastore do Apache Spark Hive. Para saber mais, consulte Parâmetros de trabalho do 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()
```

Exemplo: ler uma tabela do Hudi do Amazon S3 usando o AWS Glue Data Catalog

Este exemplo lê a tabela Hudi que você criou no Exemplo: escrever uma tabela Hudi no Amazon S3 e registrá-la no AWS Glue Data Catalog do Amazon S3.

> **Note**
>
> Este exemplo exige que você defina o parâmetro de trabalho `--enable-glue-datacatalog` para usar o AWS Glue Data Catalog como metastore do Apache Spark Hive. Para saber mais, consulte Parâmetros de trabalho do AWS Glue.

Python

Para este exemplo, use o 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

Neste exemplo, use o 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()
  }
}
```

Exemplo: atualizar e inserir um **`DataFrame`** a em uma tabela do Hudi no Amazon S3

Este exemplo usa o AWS Glue Data Catalog para inserir um DataFrame na tabela do Hudi que você criou em Exemplo: escrever uma tabela Hudi no Amazon S3 e registrá-la no AWS Glue Data Catalog.

> **Note**
>
> Este exemplo exige que você defina o parâmetro de trabalho `--enable-glue-datacatalog` para usar o AWS Glue Data Catalog como metastore do Apache Spark Hive. Para saber mais, consulte Parâmetros de trabalho do AWS Glue.

Python

Para este exemplo, use o 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

Neste exemplo, use o 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)
  }
}
```

Exemplo: ler uma tabela do Hudi do Amazon S3 usando o Spark

Este exemplo lê uma tabela do Hudi do Amazon S3 usando a API do 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/>")
```

Exemplo: gravar uma tabela no Amazon S3 usando o Spark

Este exemplo grava uma tabela no Amazon S3 usando o 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/>")
```

Exemplo: ler e escrever na tabela do Hudi com o controle de permissão do Lake Formation

Este exemplo lê e escreve uma tabela do Hudi com o controle de permissão do Lake Formation

1. Criar uma tabela do Hudi e registrá-la no Lake Formation
   a. Para habilitar o controle de permissão do Lake Formation, primeiro será necessário registrar o caminho da tabela do Amazon S3 no Lake Formation. Para obter mais informações, consulte Registering an Amazon S3 location (Registrar um local do Amazon S3). Você pode registrá-lo no console do Lake Formation ou usando a AWS CLI:

   ```
   aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-
   folder> --use-service-linked-role --region <REGION>
   ```

   Depois de registrar uma localização no Amazon S3, qualquer tabela do AWS Glue apontando para a localização (ou qualquer uma de suas localizações secundárias) retornará o valor do parâmetro `IsRegisteredWithLakeFormation` como verdadeiro na chamada `GetTable`.

   b. Crie uma tabela do Hudi que aponte para o caminho registrado do Amazon S3 por meio da API de dataframe do 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 permissão do Lake Formation para o perfil do IAM do trabalho do AWS Glue. Você pode conceder permissões no console do Lake Formation ou usando a AWS CLI. Para obter mais informações, consulte Conceder permissões de banco de dados usando o console do Lake Formation e o método de recurso nomeado
3. Leia a tabela do Hudi registrada no Lake Formation. O código é o mesmo que o da leitura de uma tabela do Hudi não registrada. Observe que o perfil do IAM do trabalho do AWS Glue precisa ter a permissão SELECT para que a leitura seja bem-sucedida.

```
val dataFrame = glueContext.getCatalogSource(
     database = "<your_database_name>",
     tableName = "<your_table_name>"
   ).getDataFrame()
```

4. Escreva em uma tabela do Hudi registrada no Lake Formation. O código é o mesmo que o da escrita em uma tabela do Hudi não registrada. Observe que o perfil do IAM do trabalho do AWS Glue precisa ter a permissão SUPER para que a escrita seja bem-sucedida.

```
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)
```

Usar a estrutura Delta Lake no AWS Glue

O AWS Glue 3.0 e versões posteriores são compatíveis com a estrutura Linux Foundation Delta Lake. Delta Lake é uma estrutura de armazenamento de data lake de código aberto que ajuda você a realizar transações ACID, escalar o tratamento de metadados e unificar o streaming e o processamento de dados em lote. Este tópico aborda os recursos disponíveis para usar dados no AWS Glue ao transportar ou armazenar dados em uma tabela do Delta Lake. Para saber mais sobre o Delta Lake, consulte a documentação oficial do Delta Lake.

Você pode usar o AWS Glue para realizar operações de leitura e gravação em tabelas do Delta Lake no Amazon S3 ou trabalhar com tabelas do Delta Lake usando o AWS Glue Data Catalog. Operações adicionais, como inserção, atualização e leitura e gravação em lote de tabelas, também são compatíveis. Quando você usa tabelas do Delta Lake, também tem a opção de usar métodos da biblioteca do Delta Lake Python, como `DeltaTable.forPath`. Para obter mais informações sobre a biblioteca de Python do Delta Lake, consulte a documentação de Python do Delta Lake.

A tabela a seguir lista a versão do Delta Lake incluída em cada versão do AWS Glue.

Versão do AWS Glue	Versão Delta Lake compatível
4,0	2.1.0
3.0	1.0.0

Para saber mais sobre as estruturas de data lake compatíveis com o AWS Glue, consulte Usar estruturas de data lake com trabalhos do AWS Glue ETL.

Habilitar o Delta Lake para AWS Glue

Para habilitar o Delta Lake para AWS Glue, faça as seguintes tarefas:

- Especifique `delta` como um valor para o parâmetro de trabalho `--datalake-formats`. Para ter mais informações, consulte Parâmetros de trabalho do AWS Glue.

- Crie uma chave denominada `--conf` para o trabalho do AWS Glue e defina-a com o seguinte valor. Ou então, você pode definir a configuração a seguir usando `SparkConf` no script. Essas configurações ajudam o Apache Spark a lidar corretamente com tabelas do 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
```

- O suporte à permissão do Lake Formation para tabelas do Delta está habilitado por padrão para o AWS Glue 4.0. Nenhuma configuração adicional é necessária para leitura/escrita em tabelas do Delta registradas no Lake Formation. Para ler uma tabela do Delta registrada, o perfil do IAM do trabalho do AWS Glue deve ter a permissão SELECT. Para escrever em uma tabela do Delta registrada, o perfil do IAM do trabalho do AWS Glue deve ter a permissão SUPER. Para saber mais sobre como gerenciar as permissões do Lake Formation, consulte Conceder e revogar permissões nos recursos do Catálogo de Dados.

Usar uma versão diferente do Delta Lake

Para usar uma versão do Delta Lake que não seja compatível com o AWS Glue, especifique seus próprios arquivos JAR do Delta Lake usando o parâmetro de trabalho `--extra-jars`. Não inclua `delta` como um valor para o parâmetro de trabalho `--datalake-formats`. Para usar a biblioteca do Delta Lake Python nesse caso, você deve especificar os arquivos JAR da biblioteca usando o parâmetro de trabalho `--extra-py-files`. A biblioteca do Python vem empacotada nos arquivos JAR do Delta Lake.

Exemplo: escrever uma tabela Delta Lake no Amazon S3 e registrá-la no AWS Glue Data Catalog

O script do AWS Glue ETL a seguir demonstra como gravar uma tabela do Delta Lake no Amazon S3 e registrá-la no AWS Glue Data Catalog.

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>")
```

Exemplo: ler uma tabela do Delta Lake do Amazon S3 usando o AWS Glue Data Catalog

O script do AWS Glue ETL a seguir lê a tabela do Delta Lake que você criou em Exemplo: escrever uma tabela Delta Lake no Amazon S3 e registrá-la no AWS Glue Data Catalog.

Python

Neste exemplo, use o 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

Neste exemplo, use o 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()
  }
}
```

Exemplo: inserir um **`DataFrame`** em uma tabela do Delta Lake no Amazon S3 usando o AWS Glue Data Catalog

Este exemplo insere dados na tabela do Delta Lake que você criou em Exemplo: escrever uma tabela Delta Lake no Amazon S3 e registrá-la no AWS Glue Data Catalog.

> **Note**
>
> Este exemplo exige que você defina o parâmetro de trabalho `--enable-glue-datacatalog` para usar o AWS Glue Data Catalog como metastore do Apache Spark Hive. Para saber mais, consulte Parâmetros de trabalho do AWS Glue.

Python

Neste exemplo, use o 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

Neste exemplo, use o 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)
  }
}
```

Exemplo: ler uma tabela do Delta Lake do Amazon S3 usando a API do Spark

Este exemplo lê uma tabela do Delta Lake do Amazon S3 usando a API do 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/>")
```

Exemplo: gravar uma tabela do Delta Lake no Amazon S3 usando o Spark

Exemplo: grava uma tabela do Delta Lake no Amazon S3 usando o 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/>")
```

Exemplo: ler e escrever na tabela do Delta Lake com o controle de permissão do Lake Formation

Este exemplo lê e escreve uma tabela do Delta Lake com o controle de permissão do Lake Formation.

1. Criar uma tabela do Delta e registrá-la no Lake Formation
   a. Para habilitar o controle de permissão do Lake Formation, primeiro será necessário registrar o caminho da tabela do Amazon S3 no Lake Formation. Para obter mais informações, consulte Registering an Amazon S3 location (Registrar um local do Amazon S3). Você pode registrá-lo no console do Lake Formation ou usando a AWS CLI:

```
aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-
folder> --use-service-linked-role --region <REGION>
```

Depois de registrar uma localização no Amazon S3, qualquer tabela do AWS Glue apontando para a localização (ou qualquer uma de suas localizações secundárias) retornará o valor do parâmetro `IsRegisteredWithLakeFormation` como verdadeiro na chamada `GetTable`.

b. Crie uma tabela do Delta que aponte para o caminho registrado do Amazon S3 por meio do Spark:

> **Note**
>
> Os exemplos a seguir foram criados em Python.

```
dataFrame.write \
 .format("delta") \
 .mode("overwrite") \
 .partitionBy("<your_partitionkey_field>") \
 .save("s3://<the_s3_path>")
```

Depois que os dados forem gravados no Amazon S3, use o crawler do AWS Glue para criar uma nova tabela de catálogo do Delta. Para obter mais informações, consulte Introdução ao suporte a tabelas nativas do Delta Lake com crawlers do AWS Glue.

Você também pode criar a tabela manualmente por meio da API `CreateTable` do AWS Glue.

2. Conceda permissão do Lake Formation para o perfil do IAM do trabalho do AWS Glue. Você pode conceder permissões no console do Lake Formation ou usando a AWS CLI. Para obter mais informações, consulte Conceder permissões de banco de dados usando o console do Lake Formation e o método de recurso nomeado
3. Leia a tabela do Delta registrada no Lake Formation. O código é o mesmo que o da leitura de uma tabela do Delta não registrada. Observe que o perfil do IAM do trabalho do AWS Glue precisa ter a permissão SELECT para que a leitura seja bem-sucedida.

```
# 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. Escreva em uma tabela do Delta registrada no Lake Formation. O código é o mesmo que o da escrita em uma tabela do Delta não registrada. Observe que o perfil do IAM do trabalho do AWS Glue precisa ter a permissão SUPER para que a escrita seja bem-sucedida.

   Por padrão, o AWS Glue usa Append como saveMode. Você pode alterá-lo configurando a opção saveMode em `additional_options`. Para obter informações sobre o suporte ao saveMode em tabelas do Delta, consulte Escrever em uma tabela.

```
glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Usar a estrutura Iceberg no AWS Glue

O AWS Glue 3.0 e versões posteriores são compatíveis com a estrutura Apache Iceberg para data lakes. O Iceberg fornece um formato de tabela de alta performance que funciona exatamente como uma tabela SQL. Este tópico aborda os recursos disponíveis para usar dados no AWS Glue ao transportar ou armazenar dados em uma tabela do Iceberg. Para saber mais sobre o Iceberg, consulte a documentação do Apache Iceberg.

Você pode usar o AWS Glue para realizar operações de leitura e gravação em tabelas do Iceberg no Amazon S3 ou trabalhar com tabelas do Iceberg usando o AWS Glue Data Catalog. Operações adicionais, incluindo inserção, atualização e todas as consultas do Spark e gravações do Spark também são compatíveis.

> **Note**
>
> `ALTER TABLE … RENAME TO` não está disponível para o Apache Iceberg 0.13.1 para AWS Glue 3.0.

A tabela a seguir lista a versão do Iceberg incluída em cada versão do AWS Glue.

Versão do AWS Glue	Versão compatível do Iceberg
4,0	1.0.0
3.0	0.13.1

Para saber mais sobre as estruturas de data lake compatíveis com o AWS Glue, consulte Usar estruturas de data lake com trabalhos do AWS Glue ETL.

Habilitar a estrutura Iceberg

Para habilitar o Iceberg para AWS Glue, faça as seguintes tarefas:

- Especifique `iceberg` como um valor para o parâmetro de trabalho `--datalake-formats`. Para ter mais informações, consulte Parâmetros de trabalho do AWS Glue.
- Crie uma chave denominada `--conf` para o trabalho do AWS Glue e defina-a com o seguinte valor. Ou então, você pode definir a configuração a seguir usando `SparkConf` no script. Essas configurações ajudam o Apache Spark a lidar corretamente com as tabelas do 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
```

  Se você estiver lendo ou gravando em tabelas do Iceberg registradas no Lake Formation, adicione a configuração a seguir para ativar o suporte ao Lake Formation. Observe que somente o AWS Glue 4.0 é compatível com tabelas do Iceberg registradas no Lake Formation:

```
--conf spark.sql.catalog.glue_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.glue_catalog.glue.id=<table-catalog-id>
```

  Se você usar o AWS Glue 3.0 com o Iceberg 0.13.1, deverá definir as seguintes configurações adicionais para usar o gerenciador de bloqueios do Amazon DynamoDB para garantir a transação atômica. AWS O Glue 4.0 usa o bloqueio positivo por padrão. Para obter mais informações, consulte Iceberg AWS Integrations na documentação oficial do 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 uma versão diferente do Iceberg

Para usar uma versão do Iceberg que não é compatível com o AWS Glue, especifique seus próprios arquivos JAR do Iceberg usando o parâmetro de trabalho `--extra-jars`. Não inclua `iceberg` como um valor para o parâmetro `--datalake-formats`.

Habilitar criptografia para tabelas do Iceberg

> **Note**
>
> As tabelas do Iceberg têm seus próprios mecanismos para habilitar a criptografia no lado do servidor. Você deve habilitar essa configuração além da configuração de segurança do AWS Glue.

Para habilitar a criptografia do lado do servidor nas tabelas do Iceberg, consulte as orientações na documentação do Iceberg.

Exemplo: escrever uma tabela do Iceberg no Amazon S3 e registrá-la no AWS Glue Data Catalog

Esse exemplo de script demonstra como escrever uma tabela Iceberg no Amazon S3. O exemplo usa aso integrações da AWS no Iceberg para registrar a tabela no AWS Glue Data Catalog.

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, você pode escrever uma tabela do Iceberg no Amazon S3 e no catálogo de dados usando os métodos do Spark.

Pré-requisitos: você precisará provisionar um catálogo para uso da biblioteca do Iceberg. Ao usar o catálogo de dados do AWS Glue, o AWS Glue simplifica essa operação. O catálogo de dados do AWS Glue é pré-configurado para ser usado pelas bibliotecas do Spark como `glue_catalog`. As tabelas do catálogo de dados são identificadas por um*`databaseName`* e um *`tableName`*. Para obter mais informações sobre o catálogo de dados do AWSGlue, consulte Descoberta e catalogação de dados.

Se você não estiver usando o catálogo de dados do AWS Glue, precisará provisionar um catálogo por meio das APIs do Spark. Para obter mais informações, consulte Configuração do Spark na documentação do Iceberg.

Este exemplo grava uma tabela Iceberg no Amazon S3 e no Data Catalog usando 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()
```

Exemplo: ler uma tabela do Iceberg do Amazon S3 usando o AWS Glue Data Catalog

Este exemplo lê a tabela do Iceberg que você criou em Exemplo: escrever uma tabela do Iceberg no Amazon S3 e registrá-la no AWS Glue Data Catalog.

Python

Para este exemplo, use o 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

Neste exemplo, use o 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()
  }
}
```

Exemplo: inserir um **`DataFrame`** em um tabela do Iceberg no Amazon S3 usando o AWS Glue Data Catalog

Este exemplo insere dados na tabela do Iceberg que você criou em Exemplo: escrever uma tabela do Iceberg no Amazon S3 e registrá-la no AWS Glue Data Catalog.

> **Note**
>
> Este exemplo exige que você defina o parâmetro de trabalho `--enable-glue-datacatalog` para usar o AWS Glue Data Catalog como metastore do Apache Spark Hive. Para saber mais, consulte Parâmetros de trabalho do AWS Glue.

Python

Para este exemplo, use o 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

Neste exemplo, use o 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)
  }
}
```

Exemplo: ler uma tabela do Iceberg do Amazon S3 usando o Spark

Pré-requisitos: você precisará provisionar um catálogo para uso da biblioteca do Iceberg. Ao usar o catálogo de dados do AWS Glue, o AWS Glue simplifica essa operação. O catálogo de dados do AWS Glue é pré-configurado para ser usado pelas bibliotecas do Spark como `glue_catalog`. As tabelas do catálogo de dados são identificadas por um*`databaseName`* e um *`tableName`*. Para obter mais informações sobre o catálogo de dados do AWSGlue, consulte Descoberta e catalogação de dados.

Se você não estiver usando o catálogo de dados do AWS Glue, precisará provisionar um catálogo por meio das APIs do Spark. Para obter mais informações, consulte Configuração do Spark na documentação do Iceberg.

Este exemplo lê uma tabela do Iceberg no Amazon S3 do catálogo de dados usando o 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")
```

Exemplo: ler e escrever na tabela do Iceberg com o controle de permissão do Lake Formation

Este exemplo lê e escreve uma tabela do Iceberg com o controle de permissão do Lake Formation.

1. Crie uma tabela do Iceberg e registrá-la no Lake Formation:
   a. Para habilitar o controle de permissão do Lake Formation, primeiro será necessário registrar o caminho da tabela do Amazon S3 no Lake Formation. Para obter mais informações, consulte Registering an Amazon S3 location (Registrar um local do Amazon S3). Você pode registrá-lo no console do Lake Formation ou usando a AWS CLI:

   ```
   aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-folder> --use-service-linked-role --region <REGION>
   ```

   Depois de registrar uma localização no Amazon S3, qualquer tabela do AWS Glue apontando para a localização (ou qualquer uma de suas localizações secundárias) retornará o valor do parâmetro `IsRegisteredWithLakeFormation` como verdadeiro na chamada `GetTable`.
   b. Crie uma tabela do Iceberg que aponte para o caminho registrado por meio do Spark:

> **Note**
>
> Os exemplos a seguir foram criados em 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)
```

Você também pode criar a tabela manualmente por meio da API `CreateTable` do AWS Glue. Para obter mais informações, consulte Criar tabelas do Apache Iceberg.

2. Conceda permissão do Lake Formation para o perfil do IAM do trabalho. Você pode conceder permissões no console do Lake Formation ou usando a AWS CLI. Para obter mais informações, consulte: https://docs.aws.amazon.com/lake-formation/latest/dg/granting-table-permissions.html
3. Leia uma tabela do Iceberg registrada no Lake Formation. O código é o mesmo que o da leitura de uma tabela do Iceberg não registrada. Observe que o perfil do IAM do trabalho do AWS Glue precisa ter a permissão SELECT para que a leitura seja bem-sucedida.

```
# 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. Escreva em uma tabela do Iceberg registrada no Lake Formation. O código é o mesmo que o da escrita em uma tabela do Iceberg não registrada. Observe que o perfil do IAM do trabalho do AWS Glue precisa ter a permissão SUPER para que a escrita seja bem-sucedida.

```
glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Referência de configuração compartilhada

Você pode usar os valores de `format_options` a seguir com qualquer tipo de formato.

- `attachFilename`: uma string no formato apropriado para ser usada como nome de coluna. Se você fornecer essa opção, o nome do arquivo de origem do registro será anexado ao registro. O valor do parâmetro será usado como nome da coluna.
- `attachTimestamp`: uma string no formato apropriado para ser usada como nome de coluna. Se você fornecer essa opção, a hora da modificação do arquivo de origem do registro será anexado ao registro. O valor do parâmetro será usado como nome da coluna.

Compatibilidade do AWS Glue Data Catalog com trabalhos do Spark SQL

O AWS Glue Data Catalog é um catálogo compatível com o metastore do Apache Hive. Você pode configurar seus trabalhos e endpoints de desenvolvimento do AWS Glue para usar o catálogo de dados como um metastore externo do Apache Hive. Depois, você pode executar diretamente as consultas do Apache Spark SQL nas tabelas armazenados no catálogo de dados. Os quadros dinâmicos do AWS Glue se integram ao catálogo de dados por padrão. No entanto, com esse recurso, os trabalhos do Spark SQL podem começar a usar o Data Catalog como um metastore externo do Hive.

Este recurso requer acesso da rede ao endpoint da API do AWS Glue. Para trabalhos do AWS Glue com conexões localizadas em sub-redes privadas, você deve configurar um endpoint da VPC ou um gateway NAT para fornecer acesso à rede. Para obter mais informações sobre a configuração do endpoint da VPC, consulte Configurar o acesso de rede aos armazenamentos de dados. Para criar um gateway NAT, consulte Gateways NAT no Manual do usuário da Amazon VPC.

Você pode configurar tarefas e endpoints de desenvolvimento do AWS Glue adicionando o argumento `"--enable-glue-datacatalog": ""` aos argumentos da tarefa e do endpoint de desenvolvimento, respectivamente. Enviar esse argumento define determinadas configurações

no Spark, que permitem o acesso ao Data Catalog como um metastore externo do Hive. Também habilita o suporte ao Hive no objeto `SparkSession` criado na tarefa ou no endpoint de desenvolvimento do AWS Glue.

Para habilitar o acesso ao Data Catalog, marque a caixa de seleção Usar o AWS Glue Data Catalog como o metastore Hive no grupo Opções de catálogo na página Adicionar trabalho ou Adicionar endpoint no console. Observe que a função do IAM usada para a tarefa ou o endpoint de desenvolvimento deve ter permissões `glue:CreateDatabase`. Um banco de dados chamado "`default`" (padrão) é criado no Data Catalog, caso não exista.

Vamos analisar um exemplo de como é possível usar esse recurso em suas tarefas do Spark SQL. O exemplo a seguir assume que você rastreou o conjunto de dados de legisladores dos EUA disponível em `s3://awsglue-datasets/examples/us-legislators`.

Para serializar/desserializar dados das tabelas definidas no AWS Glue Data Catalog, o Spark SQL precisa da classe Hive SerDe para o formato definido no AWS Glue Data Catalog no classpath do trabalho do Spark.

SerDes para determinados formatos comuns são distribuídos pelo AWS Glue. Veja a seguir os links do Amazon S3 para eles:

- JSON
- XML
- Grok

Adicione o JSON SerDe como um JAR adicional para o endpoint de desenvolvimento. Para tarefas, você pode adicionar o SerDe usando o argumento `--extra-jars` no campo de argumentos. Para ter mais informações, consulte Parâmetros de trabalho do AWS Glue.

Veja aqui um exemplo de JSON de entrada para criar um endpoint de desenvolvimento com o Data Catalog habilitado para o 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"
}
```

Agora, consulte as tabelas criadas a partir do conjunto de dados dos legisladores dos EUA usando o 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|
+--------------------+---------+-----------------+
```

Se a classe SerDe para o formato não estiver disponível no caminho de classe da tarefa, ocorrerá um erro semelhante ao seguinte.

```
>>> 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 visualizar apenas os `organization_ids` distintos da tabela `memberships`, execute a seguinte consulta SQL.

```
>>> spark.sql("select distinct organization_id from memberships_json").show()
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

Se precisar fazer o mesmo com os quadros dinâmicos, execute o seguinte.

```
>>> 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...|
+--------------------+
```

Embora DynamicFrames sejam otimizados para operações de ETL, habilitar o Spark SQL para acessar o Data Catalog diretamente fornece uma maneira concisa para executar instruções SQL complexas ou aplicações de portas existentes.

Usar marcadores de trabalho

O AWS Glue para Spark usa marcadores de trabalho para rastrear dados que já foram processados. Para obter um resumo do atributo de marcadores de emprego e o que ele suporta, consulte the section called “Rastrear dados processados usando marcadores de trabalho”. Ao programar um trabalho do AWS Glue com marcadores, você tem acesso à flexibilidade indisponível em trabalhos visuais.

- Ao ler do JDBC, você pode especificar as colunas a serem usadas como chaves de marcadores no seu script do AWS Glue.
- Você pode escolher qual `transformation_ctx` aplicar a cada chamada de método.

Sempre chame *`job.init`* no início do script e *`job.commit`* no final do script com parâmetros configurados adequadamente. Essas duas funções inicializam o serviço de marcadores e atualizam a alteração de estado para o serviço. Marcadores não funcionarão sem chamá-las.

Especifique chaves de marcadores

Para fluxos de trabalho JDBC, o marcador acompanha quais linhas seu trabalho leu comparando os valores dos campos-chave com um valor marcado. Isso não é necessário nem aplicável aos fluxos de trabalho do Amazon S3. Ao escrever um script do AWS Glue sem o editor visual, você pode especificar qual coluna rastrear com marcadores. Você pode também especificar várias colunas. Lacunas na sequência de valores são permitidas ao especificar chaves de marcadores definidas pelo usuário.

> **Warning**
>
> Se forem usadas chaves de marcadores definidas pelo usuário, elas deverão aumentar ou diminuir estritamente de forma monotônica. Ao selecionar campos adicionais para uma chave composta, os campos para conceitos como “versões secundárias” ou “números de revisão” não atendem a esses critérios, pois seus valores são reutilizados em todo o seu conjunto de dados.

É possível especificar `jobBookmarkKeys` e `jobBookmarkKeysSortOrder` das seguintes maneiras:

- `create_dynamic_frame.from_catalog`: use `additional_options`.
- `create_dynamic_frame.from_options`: use `connection_options`.

Contexto de transformação

Muitos dos métodos de quadros dinâmicos PySpark do AWS Glue incluem um parâmetro opcional chamado `transformation_ctx`, que é um identificador exclusivo para a instância do operador do ETL. O parâmetro `transformation_ctx` é usado para identificar informações de estado

em um marcador de trabalho para o determinado operador. Especificamente, o AWS Glue usa `transformation_ctx` para indexar a chave para o marcador de estado.

> **Warning**
>
> `transformation_ctx` serve como a chave para pesquisar o estado do marcador para uma fonte específica em seu script. Para que o marcador funcione corretamente, você sempre deve manter a fonte e o `transformation_ctx` associado consistentes. Alterar a propriedade da fonte ou renomear `transformation_ctx` pode tornar o marcador anterior inválido e talvez a filtragem baseada em carimbo de data/hora não ofereça o resultado correto.

Para que os marcadores de trabalho funcionem corretamente, habilite o parâmetro de marcador de trabalho e defina o parâmetro `transformation_ctx`. Se você não passar o parâmetro `transformation_ctx`, os marcadores de trabalho não serão habilitados para uma tabela ou quadro dinâmico usado no método. Por exemplo, se você tiver um trabalho de ETL que lê e une duas fontes do Amazon S3, poderá optar por passar o parâmetro `transformation_ctx` somente para os métodos em que deseja habilitar marcadores. Se você redefinir o marcador de um trabalho, ele redefinirá todas as transformações associadas ao trabalho, independentemente do `transformation_ctx` usado.

Para obter mais informações sobre a classe `DynamicFrameReader`, consulte DynamicFrameReader classe. Para obter mais informações sobre as extensões PySpark, consulte Referência de extensões PySpark do AWS Glue.

Exemplos

Example

Veja a seguir um exemplo de um script gerado para uma fonte de dados do Amazon S3. As partes do script necessárias para usar marcadores de trabalho são mostradas em itálico. Para obter mais informações sobre esses elementos, consulte a API GlueContext classee a API Classe DynamicFrameWriter.

```
# 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

Veja a seguir um exemplo de um script gerado para uma fonte JDBC. A tabela de origem é uma tabela de funcionário com a coluna empno como a chave primária. Embora, por padrão, o trabalho use uma chave primária sequencial como chave de marcador se nenhuma chave de marcador for especificada, já que empno não é necessariamente sequencial (pode haver lacunas nos valores), ela não estará qualificada como chave de marcador padrão. Portanto, o script designa explicitamente empno como a chave de marcador. Essa parte do código é mostrada em itálico.

```
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()
```

Usando a detecção de dados confidenciais fora do AWS Glue Studio

AWS Glue O Studio permite que você detecte dados confidenciais, no entanto, você também pode usar a funcionalidade de Detecção de dados confidenciais fora do AWS Glue Studio.

Para obter uma lista dos tipos de dados confidenciais gerenciados, consulte Tipos de dados gerenciados.

Detecção de dados confidenciais Detecção usando tipos AWS gerenciados de PII

AWS Glue fornece duas APIs em uma tarefa de AWS Glue ETL. Essas APIs são `detect()` e `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")
```

Você pode usar a `detect()` API para identificar tipos de PII AWS gerenciadas e tipos de entidades personalizadas. Uma nova coluna é criada automaticamente com o resultado da detecção. A API `classifyColumns()` retorna um mapa em que as chaves são nomes de colunas e os valores são uma lista dos tipos de entidades detectados. `SampleFraction` indica a fração dos dados que deve ser amostrada ao escanear entidades de PII, enquanto `ThresholdFraction` indica a fração de dados que deve ser atendida para que uma coluna seja identificada como dados de PII.

Detecção em nível de linha

No exemplo, o trabalho está realizando as seguintes ações usando as APIs `classifyColumns()` e `detect()`:

- lendo dados de um Amazon S3 bucket e os transforma em um DynamicFrame
- detectar instâncias de "e-mail" e "cartão de crédito" no dynamicFrame
- retornar um dynamicFrame com os valores originais mais uma coluna que inclui o resultado da detecção para cada linha
- gravando o DynamicFrame retornado em outro caminho 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()
   }
 }
```

Detecção em nível de linha com ações refinadas

No exemplo, o trabalho está realizando as seguintes ações usando as APIs `detect()`:

- lendo dados de um bucket do Amazon S3 e transformando-os em um dynamicFrame;
- detectando tipos de dados confidenciais para “USA_PTIN”, “BANK_ACCOUNT”, “USA_SSN”, “USA_PASSPORT_NUMBER” e “PHONE_NUMBER” no DynamicFrame;
- retornando um dynamicFrame com os valores mascarados modificados mais uma coluna que inclui o resultado da detecção para cada linha;
- gravando o dynamicFrame retornado em outro caminho do Amazon S3

Em contraste com a API `detect()` acima, isso usa ações refinadas para detectar tipos de entidades. Para ter mais informações, consulte Parâmetros de detecção para 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()
  }
```

```
}
```

Detecção em nível de coluna

No exemplo, o trabalho está realizando as seguintes ações usando as APIs `classifyColumns()`:

- lendo dados de um bucket do Amazon S3 e transformando-os em um dynamicFrame;
- detectando instâncias de "Email" e "Credit Card" no dynamicFrame;
- definindo parâmetros para amostrar 100% da coluna e marcar uma entidade como detectada se ela estiver em 10% das células e apresentar sensibilidade "BAIXA";
- retornando um mapa em que as chaves são os nomes das colunas e os valores são uma lista dos tipos de entidades detectados
- gravando o dynamicFrame retornado em outro caminho do 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()
  }
}
```

Detecção de dados confidenciais Detecção usando tipos de AWS CustomEntityType PII

Você pode definir entidades personalizadas por meio do AWS Studio. No entanto, para usar esse recurso fora do AWS Studio, você precisa primeiro definir os tipos de entidade personalizados e, em seguida, adicionar os tipos de entidade personalizados definidos à lista de`entityTypesToDetect`.

Se você tiver tipos de dados confidenciais específicos em seus dados (como “ID do funcionário”), poderá criar entidades personalizadas chamando a API `CreateCustomEntityType()`.
O exemplo a seguir define o tipo de entidade personalizada 'EMPLOYEE_ID' para a API `CreateCustomEntityType()` com os parâmetros de solicitação:

```
  {
      "name": "EMPLOYEE_ID",
      "regexString": "\d{4}-\d{3}",
      "contextWords": ["employee"]
  }
```

Em seguida, modifique o trabalho para usar o novo tipo de dados confidenciais personalizados adicionando o tipo de entidade personalizada (EMPLOYEE_ID) à 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**
>
> Se um tipo de dados confidenciais personalizados for definido com o mesmo nome de um tipo de entidade gerenciada existente, o tipo de dados confidenciais personalizados terá precedência e substituirá a lógica do tipo de entidade gerenciada.

Parâmetros de detecção para uso de **`detect()`**

Esse método é usado para detectar entidades em um DynamicFrame. Ele retorna um novo DataFrame com valores originais e uma coluna outputColumnName adicional com metadados de detecção de PII. O mascaramento personalizado pode ser feito depois que isso DynamicFrame for retornado no AWS Glue script, ou a API detect () com ações refinadas pode ser usada em seu lugar.

```
detect(frame: DynamicFrame,
       entityTypesToDetect: Seq[String],
       outputColumnName: String = "DetectedEntities",
       detectionSensitivity: String = "LOW"): DynamicFrame
```

Parâmetros:

- frame — (tipo:`DynamicFrame`) A entrada DynamicFrame contendo os dados a serem processados.
- entityTypesToDetectar — (tipo:`[Seq[String]`) Lista de tipos de entidades a serem detectadas. Esses podem ser tipos de entidade gerenciados ou tipos de entidade personalizados.
- outputColumnName— (type:`String`, default: "DetectedEntities“) O nome da coluna em que as entidades detectadas serão armazenadas. Se não for fornecido, o nome padrão da coluna será "DetectedEntities”.
- detectionSensitivity (tipo:`String`, opções: "BAIXA" ou "ALTA", padrão: "BAIXA"): especifica a sensibilidade do processo de detecção. As opções válidas são "BAIXA" ou "ALTA". Se essa propriedade não for definida, a sensibilidade padrão será definida como "BAIXA".

Configurações do `outputColumnName`:

O nome da coluna em que as entidades detectadas serão armazenadas. Se não for fornecido, o nome padrão da coluna será "DetectedEntities”. Para cada linha na coluna de saída, a coluna suplementar inclui um mapa do nome da coluna para os metadados da entidade detectada com os seguintes pares de chave-valor:

- entityType: o tipo da entidade detectada.
- start: a posição inicial da entidade detectada nos dados originais.
- end: a posição final da entidade detectada nos dados originais.

- actionUsed: a ação executada na entidade detectada (por exemplo, “DETECT”, “REDACT”, “PARTIAL_REDACT”, “SHA256_HASH”).

Exemplo:

```
{
   "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
         }
      ]
   }
}
```

Parâmetros de detecção para **`detect()`** com ações refinadas

Esse método é usado para detectar entidades em um DynamicFrame usando parâmetros especificados. Ele retorna um novo DataFrame com os valores originais substituídos por dados confidenciais mascarados e uma coluna adicional com `outputColumnName` metadados de detecção de PII.

```
detect(frame: DynamicFrame,
```

```
        detectionParameters: JsonOptions,
        outputColumnName: String = "DetectedEntities",
        detectionSensitivity: String = "LOW"): DynamicFrame
```

Parâmetros:

- frame — (type:`DynamicFrame`): A entrada DynamicFrame contendo os dados a serem processados.
- detectionParameters (tipo:`JsonOptions`): opções JSON que especificam parâmetros para o processo de detecção.
- outputColumnName— (type:`String`, default: "DetectedEntities“): O nome da coluna em que as entidades detectadas serão armazenadas. Se não for fornecido, o nome padrão da coluna será "DetectedEntities”.
- detectionSensitivity (tipo:`String`, opções: "BAIXA" ou "ALTA", padrão: "BAIXA"): especifica a sensibilidade do processo de detecção. As opções válidas são "BAIXA" ou "ALTA". Se essa propriedade não for definida, a sensibilidade padrão será definida como "BAIXA".

configurações `detectionParameters`

Se nenhuma configuração for incluída, os valores padrão serão usados.

- action (tipo:`String`, opções: "DETECT", "REDACT", "PARTIAL_REDACT", "SHA256_HASH"): especifica a ação a ser executada na entidade. Obrigatório. Observe que as ações que executam o mascaramento (todas com exceção de "DETECT") só podem realizar uma ação por coluna. Essa é uma medida preventiva para mascarar entidades coalescidas.
- sourceColumns (tipo: `List[String]`, padrão: ["*"]): lista de nomes de colunas de origem para realizar a detecção da entidade. Por padrão, será ["*"] se não estiver presente. Aumentará `IllegalArgumentException` se um nome de coluna inválido for usado.
- sourceColumnsToExcluir — (digite:`List[String]`) Lista de nomes de colunas de origem para realizar a detecção da entidade. Use `sourceColumns` ou `sourceColumnsToExclude`. Aumentará `IllegalArgumentException` se um nome de coluna inválido for usado.
- actionOptions: opções adicionais com base na ação especificada:
  - Para "DETECT" e "SHA256_HASH", nenhuma opção é permitida.
  - Para "REDACT":
    - redactText (tipo:`String`, padrão: "*****"): texto para substituir a entidade detectada.

- Para "PARTIAL_REDACT":
  - redactChar (tipo:`String`, padrão: "*"): caractere para substituir cada caractere detectado na entidade.
  - matchPattern (tipo:`String`): padrão Regex para redação parcial. Não pode ser combinado com numLeftChars ToExclude ou`numRightCharsToExclude`.
  - numLeftCharsToExclude— (tipo:`String, integer`) Número de caracteres à esquerda a serem excluídos. Não pode ser combinado com matchPattern, mas pode ser usado com `numRightCharsToExclude`.
  - numRightCharsToExclude— (tipo:`String, integer`) Número de caracteres corretos a serem excluídos. Não pode ser combinado com matchPattern, mas pode ser usado com `numRightCharsToExclude`.

Configurações de `outputColumnName`

Veja outputColumnName as configurações

Parâmetros de detecção para **`classifyColumns()`**

Esse método é usado para detectar entidades em um DynamicFrame. Ele retorna um mapa em que as chaves são os nomes das colunas e os valores são uma lista dos tipos de entidades detectados. O mascaramento personalizado poderá ser feito depois que isso for retornado no script AWS Glue.

```
classifyColumns(frame: DynamicFrame,
                entityTypesToDetect: Seq[String],
                sampleFraction: Double = 0.1,
                thresholdFraction: Double = 0.1,
                detectionSensitivity: String = "LOW")
```

Parâmetros:

- frame — (tipo:`DynamicFrame`) A entrada DynamicFrame contendo os dados a serem processados.
- entityTypesToDetectar — (tipo:`Seq[String]`) Lista de tipos de entidades a serem detectadas. Esses podem ser tipos de entidade gerenciados ou tipos de entidade personalizados.
- sampleFraction (tipo:`Double`, padrão: 10%): a fração dos dados que será amostrada na verificação de entidades de PII.

- thresholdFraction (tipo:`Double`, padrão: 10%): a fração dos dados que deverá ser atendida para que uma coluna seja identificada como dados de PII.
- detectionSensitivity (tipo:`String`, opções: "BAIXA" ou "ALTA", padrão: "BAIXA"): especifica a sensibilidade do processo de detecção. As opções válidas são "BAIXA" ou "ALTA". Se essa propriedade não for definida, a sensibilidade padrão será definida como "BAIXA".

Tipos de dados confidenciais gerenciados

Entidades globais

Tipo de dado	Categoria	Descrição
PERSON_NAME	Universal	O nome da pessoa.
EMAIL	Pessoal	O endereço de e-mail.
IP_ADDRESS	Computador	O endereço IP
MAC_ADDRESS	Pessoal	O endereço MAC.

Tipos de dados dos EUA

Tipo de dado	Descrição
BANK_ACCOUNT	O número da conta bancária. Não específico de um país ou região, porém, apenas formatos de contas nos EUA e Canadá são detectados no momento.
CREDIT_CARD	O número do cartão de crédito.
PHONE_NUMBER	Número de telefone. Não específico de um país ou região, mas, apenas números de telefone dos EUA e Canadá são detectados no momento.

Tipo de dado	Descrição
USA_ATIN	O número de identificação fiscal de adoção dos EUA emitido pela Receita Federal.
USA_CPT_CODE	O código CPT (específico dos EUA).
USA_DEA_NUMBER	O número do DEA (específico dos EUA).
USA_DRIVING_LICENSE	O número da carteira de motorista (específico dos EUA).
USA_HCPCS_CODE	O código HCPCS (específico dos EUA).
USA_HEALTH_INSURANCE_CLAIM_NUMBER	Número da reividicação do seguro de saúde (específico dos EUA).
USA_ITIN	O ITIN (para pessoas físicas ou entidades nos EUA).
USA_MEDICARE_BENEFICIARY_IDENTIFIER	Identificação de beneficiário do Medicare (específico dos EUA).
USA_NATIONAL_DRUG_CODE	O código NDC (específico dos EUA).
USA_NATIONAL_PROVIDER_IDENTIFIER	O número identificador de prestador de serviço nacional (específico dos EUA).
USA_PASSPORT_NUMBER	O número do passaporte (para pessoas físicas dos EUA).
USA_PTIN	O número de identificação fiscal de adoção dos EUA emitido pela Receita Federal.
USA_SSN	O número da previdência social (para pessoas físicas nos EUA).

Tipos de dados da Argentina

Tipo de dado	Descrição
ARGENTINA_TAX_IDENTIFICATIO N_NUMBER	Número de identificação fiscal da Argentina. Também conhecido como CUIT ou CUIL.

Tipos de dados australianos

Tipo de dado	Descrição
AUSTRALIA_BUSINESS_NUMBER	Número de empresa australiana (ABN). Um identificador exclusivo emitido pelo Registro de Empresas Australianas (ABR) para identificar empresas para o governo e a comunidade.
AUSTRALIA_COMPANY_NUMBER	Número da empresa australiana (ACN). Identificador exclusivo emitido pela Comissão Australiana de Valores Mobiliários e Investime ntos.
AUSTRALIA_DRIVING_LICENSE	O número da carteira de motorista da Austrália.
AUSTRALIA_MEDICARE_NUMBER	Número do Medicare australiano. Identificador pessoal emitido pela Comissão Australiana de Seguro de Saúde.
NÚMERO DO PASSAPORTE AUSTRALIANO	Número do passaporte australiano.
AUSTRALIA_TAX_FILE_NUMBER	Número do arquivo fiscal da Austrália (TFN). Emitido pelo Escritório Australiano de Tributaçã o (ATO) para contribuintes (pessoas físicas, jurídicas, etc.) para transações tributárias.

Tipos de dados da Áustria

Tipo de dado	Descrição
AUSTRIA_DRIVING_LICENSE	O número da carteira de motorista (específico da Áustria).
AUSTRIA_PASSPORT_NUMBER	O número do passaporte (específico da Áustria).
AUSTRIA_SSN	O número da previdência social (para pessoas físicas da Áustria).
AUSTRIA_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Áustria).
AUSTRIA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Áustria).

Tipos de dados dos Bálcãs

Tipo de dado	Descrição
BOSNIA_UNIQUE_MASTER_CITIZEN_NUMBER	Número de cidadão mestre exclusivo (JMBG) para cidadãos da Bósnia-Herzegovina.
KOSOVO_UNIQUE_MASTER_CITIZEN_NUMBER	Número de cidadão mestre exclusivo (JMBG) para o Kosovo.
MACEDONIA_UNIQUE_MASTER_CITIZEN_NUMBER	Número de cidadão mestre exclusivo para a Macedônia.
MONTENEGRO_UNIQUE_MASTER_CITIZEN_NUMBER	Número de cidadão mestre exclusivo (JMBG) para Montenegro.
SERBIA_UNIQUE_MASTER_CITIZEN_NUMBER	Número de cidadão mestre exclusivo (JMBG) para a Sérvia.
SERBIA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Áustria).

Tipo de dado	Descrição
VOJVODINA_UNIQUE_MASTER_CITIZEN_NUMBER	Número de cidadão mestre exclusivo (JMBG) para a Voivodina.

Tipos de dados da Bélgica

Tipo de dado	Descrição
BELGIUM_DRIVING_LICENSE	O número da carteira de motorista (específico da Bélgica).
BELGIUM_NATIONAL_IDENTIFICATION_NUMBER	O número nacional belga (BNN).
BELGIUM_PASSPORT_NUMBER	O número do passaporte (específico da Bélgica).
BELGIUM_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Bélgica).
BELGIUM_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Bélgica).

Tipos de dados do Brasil

Tipo de dado	Descrição
BRAZIL_BANK_ACCOUNT	O número da conta bancária (específico do Brasil).
BRAZIL_NATIONAL_IDENTIFICATION_NUMBER	O identificador nacional (específico do Brasil).
BRAZIL_NATIONAL_REGISTRY_OF_LEGAL_ENTITIES_NUMBER	O número de identificação emitido para empresas (específico do Brasil), também conhecido como CNPJ.

Tipo de dado	Descrição
BRAZIL_NATURAL_PERSON_REGISTRY_NUMBER	Número de registro de pessoa física, também conhecido como CPF.

Tipos de dados da Bulgária

Tipo de dado	Descrição
BULGARIA_DRIVING_LICENSE	O número da carteira de motorista (específico da Bulgária).
BULGARIA_UNIFORM_CIVIL_NUMBER	Número civil unificado (EGN) que serve como número de identificação nacional.
BULGARIA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Bulgária).

Tipos de dados do Canadá

Tipo de dado	Descrição
CANADA_DRIVING_LICENSE	O número da carteira de motorista (específico do Canadá).
CANADA_GOVERNMENT_IDENTIFICATION_CARD_NUMBER	O identificador nacional (específico do Canadá).
CANADA_PASSPORT_NUMBER	O número do passaporte (específico do Canadá).
CANADA_PERMANENT_RESIDENCE_NUMBER	Número de residência permanente (número do cartão PR).
CANADA_PERSONAL_HEALTH_NUMBER	O identificador exclusivo para assistência médica (número PHN).

Tipo de dado	Descrição
CANADA_SOCIAL_INSURANCE_NUMBER	O número do seguro social (SIN) no Canadá.

Tipos de dados do Chile

Tipo de dado	Descrição
CHILE_DRIVING_LICENSE	O número da carteira de motorista (específico do Chile).
CHILE_NATIONAL_IDENTIFICATION_NUMBER	O identificador nacional do Chile, também conhecido como RUT ou RUN.

Tipos de dados da China, Hong Kong, Macau e Taiwan

Tipo de dado	Descrição
CHINA_IDENTIFICATION	O identificador da China.
CHINA_LICENSE_PLATE_NUMBER	O número da carteira de motorista (específico da China).
CHINA_MAINLAND_TRAVEL_PERMIT_ID_HONG_KONG_MACAU	A autorização de viagem ao continente para residentes de Hong Kong e Macau.
CHINA_MAINLAND_TRAVEL_PERMIT_ID_TAIWAN	A autorização de viagem ao continente para residentes de Taiwan emitida pelo Governo da República Popular da China (RPC).
CHINA_PASSPORT_NUMBER	O número do passaporte (específico da China).
CHINA_PHONE_NUMBER	O número de telefone (específico da China).
HONG_KONG_IDENTITY_CARD	O documento de identidade oficial emitido pelo Departamento de Imigração de Hong Kong.

Tipo de dado	Descrição
MACAU_RESIDENT_IDENTITY_CARD	A carteira de identidade de residente de Macau ou BIR é uma carteira de identidade oficial emitida pelo Gabinete de Serviços de Identific ação de Macau.
TAIWAN_NATIONAL_IDENTIFICAT ION_NUMBER	O identificador nacional (específico de Taiwan).
TAIWAN_PASSPORT_NUMBER	O número do passaporte (específico de Taiwan).

Tipos de dados da Colômbia

Tipo de dado	Descrição
COLOMBIA_PERSONAL_IDENTIFIC ATION_NUMBER	Identificador exclusivo atribuído aos colombian os no nascimento.
COLOMBIA_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Colômbia).

Tipos de dados da Croácia

Tipo de dado	Descrição
CROATIA_DRIVING_LICENSE	O número da carteira de motorista (específico da Croácia).
CROATIA_IDENTITY_NUMBER	O identificador nacional (específico da Croácia).
CROATIA_PASSPORT_NUMBER	O número do passaporte (específico da Croácia).

Tipo de dado	Descrição
CROATIA_PERSONAL_IDENTIFICATION_NUMBER	O número de identificação pessoal (OIB).

Tipos de dados de Chipre

Tipo de dado	Descrição
CYPRUS_DRIVING_LICENSE	O número da carteira de motorista (específico de Chipre).
CYPRUS_NATIONAL_IDENTIFICATION_NUMBER	A carteira de identidade cipriota.
CYPRUS_PASSPORT_NUMBER	O número do passaporte (específico de Chipre).
CYPRUS_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico de Chipre).
CYPRUS_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico de Chipre).

Tipos de dados da República Tcheca

Tipo de dado	Descrição
CZECHIA_DRIVING_LICENSE	O número da carteira de motorista (específico da República Tcheca).
CZECHIA_PERSONAL_IDENTIFICATION_NUMBER	O número de identificação pessoal (específico da República Tcheca).
CZECHIA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da República Tcheca).

Tipos de dados da Dinamarca

Tipo de dado	Descrição
DENMARK_DRIVING_LICENSE	O número da carteira de motorista (específico da Dinamarca).
DENMARK_PERSONAL_IDENTIFICATION_NUMBER	O número de identificação pessoal (específico da Dinamarca).
DENMARK_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Dinamarca).
DINAMARCA_VALOR_IMPOSTO_AGREGADO	Imposto sobre valor agregado (específico da Dinamarca).

Tipos de dados da Estônia

Tipo de dado	Descrição
ESTONIA_DRIVING_LICENSE	O número da carteira de motorista (específico da Estônia).
ESTONIA_PASSPORT_NUMBER	O número do passaporte (específico da Estônia).
ESTONIA_PERSONAL_IDENTIFICATION_CODE	O número de identificação pessoal (específico da Estônia).
ESTONIA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Estônia).

Tipos de dados da Finlândia

Tipo de dado	Descrição
FINLAND_DRIVING_LICENSE	O número da carteira de motorista (específico da Finlândia).
FINLAND_HEALTH_INSURANCE_NUMBER	O número do seguro saúde (específico da Finlândia).
FINLAND_NATIONAL_IDENTIFICATION_NUMBER	O número identificador de prestador de serviço nacional (específico da Finlândia).
FINLAND_PASSPORT_NUMBER	O número do passaporte (específico da Finlândia).
FINLAND_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Finlândia).

Tipos de dados da França

Tipo de dado	Descrição
FRANCE_BANK_ACCOUNT	O número da conta bancária (específico da França).
FRANCE_DRIVING_LICENSE	O número da carteira de motorista (específico da França).
FRANCE_HEALTH_INSURANCE_NUMBER	Número do seguro saúde da França.
FRANCE_INSEE_CODE	Número da previdência social, SSN ou NIR da França.
FRANCE_NATIONAL_IDENTIFICATION_NUMBER	Número de identificação nacional (CNI) da França.
FRANCE_PASSPORT_NUMBER	O número do passaporte (específico da França).

Tipo de dado	Descrição
FRANCE_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da França).
FRANCE_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da França).

Tipos de dados da Alemanha

Tipo de dado	Descrição
GERMANY_BANK_ACCOUNT	O número da conta bancária (específico da Alemanha).
GERMANY_DRIVING_LICENSE	O número da carteira de motorista (específico da Alemanha).
GERMANY_PASSPORT_NUMBER	O número do passaporte (específico da Alemanha).
GERMANY_PERSONAL_IDENTIFICATION_NUMBER	O número de identificação pessoal (específico da Alemanha).
GERMANY_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Alemanha).
GERMANY_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Alemanha).

Tipos de dados da Grécia

Tipo de dado	Descrição
GREECE_DRIVING_LICENSE	O número da carteira de motorista (específico da Grécia).

Tipo de dado	Descrição
GREECE_PASSPORT_NUMBER	O número do passaporte (específico da Grécia).
GREECE_SSN	O número da previdência social (para pessoas físicas da Grécia).
GREECE_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Grécia).
GREECE_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Grécia).

Tipos de dados da Hungria

Tipo de dado	Descrição
HUNGARY_DRIVING_LICENSE	O número da carteira de motorista (específico da Hungria).
HUNGARY_PASSPORT_NUMBER	O número do passaporte (específico da Hungria).
HUNGARY_SSN	O número da previdência social (para pessoas físicas da Hungria).
HUNGARY_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Hungria).
HUNGARY_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Hungria).

Tipos de dados da Islândia

Tipo de dado	Descrição
ICELAND_NATIONAL_IDENTIFICATION_NUMBER	O identificador nacional (específico da Islândia).
ICELAND_PASSPORT_NUMBER	O número do passaporte (específico da Islândia).
ICELAND_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Islândia).

Tipos de dados da Índia

Tipo de dado	Descrição
INDIA_AADHAAR_NUMBER	Número de identificação Aadhaar emitido pela Autoridade de Identificação Única da Índia.
INDIA_PERMANENT_ACCOUNT_NUMBER	Número da conta permanente (PAN) da Índia.

Tipos de dados da Indonésia

Tipo de dado	Descrição
INDONESIA_IDENTITY_CARD_NUMBER	O identificador nacional (específico da Indonésia).

Tipos de dados da Irlanda

Tipo de dado	Descrição
IRELAND_DRIVING_LICENSE	O número da carteira de motorista (específico da Irlanda).

Tipo de dado	Descrição
IRELAND_PASSPORT_NUMBER	O número do passaporte (específico da Irlanda).
IRELAND_PERSONAL_PUBLIC_SERVICE_NUMBER	Número de serviço público pessoal da Irlanda (PPS).
IRELAND_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Irlanda).
IMPOSTO SOBRE VALOR AGREGADO DA IRLANDA	Imposto sobre valor agregado (específico da Irlanda).

Tipos de dados de Israel

Tipo de dado	Descrição
ISRAEL_IDENTIFICATION_NUMBER	O identificador nacional (específico de Israel).

Tipos de dados da Itália

Tipo de dado	Descrição
ITALY_BANK_ACCOUNT	O número da conta bancária (específico da Itália).
ITALY_DRIVING_LICENSE	O número da carteira de motorista (específico da Itália).
ITALY_FISCAL_CODE	O número identificador, também conhecido como Codice Fiscale italiano.
ITALY_PASSPORT_NUMBER	O número do passaporte (específico da Itália).
ITALY_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Itália).

Tipos de dados do Japão

Tipo de dado	Descrição
JAPAN_BANK_ACCOUNT	Conta bancária no Japão.
JAPAN_DRIVING_LICENSE	Um número de carteira de motorista do Japão.
JAPAN_MY_NUMBER	O identificador exclusivo para cidadãos ou corporações do Japão, usado para administr ação tributária, administração de previdência social e resposta a desastres
JAPAN_PASSPORT_NUMBER	Número do passaporte do Japão.

Tipos de dados da Coreia

Tipo de dado	Descrição
KOREA_PASSPORT_NUMBER	O número do passaporte (específico da Coreia).
KOREA_RESIDENCE_REGISTRATIO N_NUMBER_FOR_CITIZENS	Número de registro de residência na Coreia para residentes.
KOREA_RESIDENCE_REGISTRATIO N_NUMBER_FOR_FOREIGNERS	Número de registro de residência na Coreia para estrangeiros.

Tipos de dados da Letônia

Tipo de dado	Descrição
LATVIA_DRIVING_LICENSE	O número da carteira de motorista (específico da Letônia).
LATVIA_PASSPORT_NUMBER	O número do passaporte (específico da Letônia).

Tipo de dado	Descrição
LATVIA_PERSONAL_IDENTIFICATION_NUMBER	O número de identificação pessoal (específico da Letônia).
LATVIA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Letônia).

Tipos de dados de Liechtenstein

Tipo de dado	Descrição
LIECHTENSTEIN_NATIONAL_IDENTIFICATION_NUMBER	O identificador nacional (específico de Liechtenstein).
LIECHTENSTEIN_PASSPORT_NUMBER	O número do passaporte (específico de Liechtenstein).
LIECHTENSTEIN_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico de Liechtenstein).

Tipos de dados da Lituânia

Tipo de dado	Descrição
LITHUANIA_DRIVING_LICENSE	O número da carteira de motorista (específico da Lituânia).
LITHUANIA_PERSONAL_IDENTIFICATION_NUMBER	O número de identificação pessoal (específico da Lituânia).
LITHUANIA_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Lituânia).
LITHUANIA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Lituânia).

Tipos de dados de Luxemburgo

Tipo de dado	Descrição
LUXEMBOURG_DRIVING_LICENSE	O número da carteira de motorista (específico de Luxemburgo).
LUXEMBOURG_NATIONAL_INDIVIDUAL_NUMBER	O identificador nacional (específico de Luxemburgo).
LUXEMBOURG_PASSPORT_NUMBER	O número do passaporte (específico de Luxemburgo).
LUXEMBOURG_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico de Luxemburgo).
LUXEMBOURG_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico de Luxemburgo).

Tipos de dados da Malásia

Tipo de dado	Descrição
MALAYSIA_MYKAD_NUMBER	O identificador nacional (específico da Malásia).
MALAYSIA_PASSPORT_NUMBER	O número do passaporte (específico da Malásia).

Tipos de dados de Malta

Tipo de dado	Descrição
MALTA_DRIVING_LICENSE	O número da carteira de motorista (específico de Malta).

Tipo de dado	Descrição
MALTA_NATIONAL_IDENTIFICATION_NUMBER	O identificador nacional (específico de Malta).
MALTA_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico de Malta).
MALTA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico de Malta).

Tipos de dados do México

Tipo de dado	Descrição
MEXICO_CLABE_NUMBER	México (número bancário CLABE (Clave Bancaria Estandarizada).
MEXICO_DRIVING_LICENSE	O número da carteira de motorista (específico do México).
MEXICO_PASSPORT_NUMBER	O número do passaporte (específico do México).
MEXICO_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico do México).
MEXICO_UNIQUE_POPULATION_REGISTRY_CODE	O código de identidade exclusivo Clave Única de Registro de Población (CURP) do México.

Tipos de dados da Holanda

Tipo de dado	Descrição
NETHERLANDS_CITIZEN_SERVICE_NUMBER	Número de cidadão holandês (BSN, burgerservicenummer).

Tipo de dado	Descrição
NETHERLANDS_DRIVING_LICENSE	O número da carteira de motorista (específico da Holanda).
NETHERLANDS_PASSPORT_NUMBER	O número do passaporte (específico da Holanda).
NETHERLANDS_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Holanda).
NETHERLANDS_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Holanda).
NETHERLANDS_BANK_ACCOUNT	O número da conta bancária (específico da Holanda).

Tipos de dados da Nova Zelândia

Tipo de dado	Descrição
NEW_ZEALAND_DRIVING_LICENSE	O número da carteira de motorista (específico da Nova Zelândia).
NEW_ZEALAND_NATIONAL_HEALTH_INDEX_NUMBER	Número do índice nacional de saúde da Nova Zelândia.
NEW_ZEALAND_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal, também conhecido como número da Receita Federal (específico da Nova Zelândia).

Tipos de dados da Noruega

Tipo de dado	Descrição
NORWAY_BIRTH_NUMBER	Número de identidade nacional norueguês.

Tipo de dado	Descrição
NORWAY_DRIVING_LICENSE	O número da carteira de motorista (específico da Noruega).
NORWAY_HEALTH_INSURANCE_NUMBER	Número do seguro saúde da Noruega.
NORWAY_NATIONAL_IDENTIFICATION_NUMBER	O número identificador de prestador de serviço nacional (específico da Noruega).
NORWAY_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Noruega).

Tipos de dados das Filipinas

Tipo de dado	Descrição
PHILIPPINES_DRIVING_LICENSE	O número da carteira de motorista (específico da Noruega).
PHILIPPINES_PASSPORT_NUMBER	O número do passaporte (específico das Filipinas).

Tipos de dados da Polônia

Tipo de dado	Descrição
POLAND_DRIVING_LICENSE	O número da carteira de motorista (específico das Filipinas).
POLAND_IDENTIFICATION_NUMBER	O identificador da Polônia.
POLAND_PASSPORT_NUMBER	O número do passaporte (específico das Filipinas).

Tipo de dado	Descrição
NÚMERO DA REGIÃO DA POLÔNIA	O número identificador REGON, também conhecido como Número de Identificação Estatística.
POLAND_SSN	O número da previdência social (para pessoas físicas da Polônia).
POLAND_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Polônia).
POLAND_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Polônia).

Tipos de dados de Portugal

Tipo de dado	Descrição
PORTUGAL_DRIVING_LICENSE	O número da carteira de motorista (específico de Portugal).
PORTUGAL_NATIONAL_IDENTIFIC ATION_NUMBER	O número identificador de prestador de serviço nacional (específico de Portugal).
NÚMERO_PASSAPORTE_PORTUGAL_BR	O número do passaporte (específico de Portugal).
PORTUGAL_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico de Portugal).
PORTUGAL_VALUE_ADDED_TAX	Imposto sobre o valor acrescentado (específico de Portugal).

Tipos de dados da Romênia

Tipo de dado	Descrição
ROMANIA_DRIVING_LICENSE	O número da carteira de motorista (específico da Romênia).
ROMANIA_NUMERICAL_PERSONAL_CODE	O número de identificação pessoal (específico da Romênia).
ROMANIA_PASSPORT_NUMBER	O número do passaporte (específico da Romênia).
ROMANIA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Romênia).

Tipos de dados de Singapura

Tipo de dado	Descrição
SINGAPORE_DRIVING_LICENSE	O número da carteira de motorista (específico de Singapura).
SINGAPORE_NATIONAL_REGISTRY_IDENTIFICATION_NUMBER	A carteira de identidade de registro nacional de Singapura.
SINGAPORE_PASSPORT_NUMBER	O número do passaporte (específico de Singapura).
SINGAPORE_UNIQUE_ENTITY_NUMBER	O número identificador exclusivo (UEN) de Singapura.

Tipos de dados da Eslováquia

Tipo de dado	Descrição
SLOVAKIA_DRIVING_LICENSE	O número da carteira de motorista (específico da Eslováquia).

Tipo de dado	Descrição
SLOVAKIA_NATIONAL_IDENTIFICATION_NUMBER	O número identificador de prestador de serviço nacional (específico da Eslováquia).
SLOVAKIA_PASSPORT_NUMBER	O número do passaporte (específico da Eslováquia).
SLOVAKIA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Eslováquia).

Tipos de dados da Eslovênia

Tipo de dado	Descrição
SLOVENIA_DRIVING_LICENSE	O número da carteira de motorista (específico da Eslovênia).
SLOVENIA_PASSPORT_NUMBER	O número do passaporte (específico da Eslovênia).
SLOVENIA_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal (específico da Eslovênia).
SLOVENIA_UNIQUE_MASTER_CITIZEN_NUMBER	Número de cidadão mestre exclusivo (JMBG) para cidadãos da Eslovênia.
SLOVENIA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Eslovênia).

Tipos de dados da África do Sul

Tipo de dado	Descrição
SOUTH_AFRICA_PERSONAL_IDENTIFICATION_NUMBER	O número de identificação pessoal (específico da África do Sul).

Tipos de dados da Espanha

Tipo de dado	Descrição
SPAIN_BANK_ACCOUNT	O número da conta bancária (específico da Espanha).
SPAIN_DNI	O cartão de identidade nacional (Documento Nacional de Identidad) da Espanha.
SPAIN_DRIVING_LICENSE	O número da carteira de motorista (específico da Espanha).
SPAIN_NIE	O número de identidade de estrangeiro (específico da Espanha), também conhecido como NIE.
SPAIN_NIF	Número de identificação fiscal (específico da Espanha), também conhecido como NIF.
SPAIN_PASSPORT_NUMBER	O número do passaporte (específico da Espanha).
SPAIN_SSN	O número da previdência social (para pessoas físicas da Espanha).
SPAIN_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Espanha).

Tipos de dados do Sri Lanka

Tipo de dado	Descrição
SRI_LANKA_NATIONAL_IDENTIFICATION_NUMBER	O identificador nacional (específico do Sri Lanka).

Tipos de dados da Suécia

Tipo de dado	Descrição
CARTEIRA DE HABILITAÇÃO SUECA	O número da carteira de motorista (específico da Suécia).
SWEDEN_PASSPORT_NUMBER	O número do passaporte (específico da Suécia).
SWEDEN_PERSONAL_IDENTIFICATION_NUMBER	O número identificador de prestador de serviço nacional (específico da Suécia).
SWEDEN_TAX_IDENTIFICATION_NUMBER	Número de identificação fiscal da Suécia (personnummer).
SWEDEN_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Suécia).

Tipos de dados da Suíça

Tipo de dado	Descrição
SWITZERLAND_AHV	O número da previdência social (para pessoas físicas da Suíça).
SWITZERLAND_HEALTH_INSURANCE_NUMBER	Número do seguro saúde suíço.
SWITZERLAND_PASSPORT_NUMBER	O número do passaporte (específico da Suíça).
SWITZERLAND_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Suíça).

Tipos de dados da Tailândia

Tipo de dado	Descrição
THAILAND_PASSPORT_NUMBER	O número do passaporte (específico da Tailândia).
THAILAND_PERSONAL_IDENTIFICATION_NUMBER	O número de identificação pessoal (específico da Tailândia).

Tipos de dados da Turquia

Tipo de dado	Descrição
TURKEY_NATIONAL_IDENTIFICATION_NUMBER	O número didentificador de prestador de serviço nacional (específico da Turquia).
TURKEY_PASSPORT_NUMBER	O número do passaporte (específico da Turquia).
TURKEY_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Turquia).

Tipos de dados da Ucrânia

Tipo de dado	Descrição
UKRAINE_INDIVIDUAL_IDENTIFICATION_NUMBER	O identificador exclusivo (específico da Ucrânia).
UKRAINE_PASSPORT_NUMBER_DOMESTIC	O número do passaporte nacional (específico da Ucrânia).
UKRAINE_PASSPORT_NUMBER_INTERNATIONAL	O número do passaporte internacional (específico da Ucrânia).

Tipos de dados dos Emirados Árabes Unidos (EAU)

Tipo de dado	Descrição
UNITED_ARAB_EMIRATES_PERSONAL_NUMBER	O número de identificação pessoal (específico dos Emirados Árabes Unidos).

Tipos de dados do Reino Unido

Tipo de dado	Descrição
UK_BANK_ACCOUNT	Conta bancária do Reino Unido (Reino Unido).
UK_BANK_SORT_CODE	Código de classificação bancária do Reino Unido (Reino Unido). Os códigos Sore são códigos bancários usados para rotear transferências de dinheiro entre bancos em seus respectivos países por meio de suas respectivas organizações de compensação.
UK_DRIVING_LICENSE	O número da carteira de motorista do Reino Unido da Grã-Bretanha e Irlanda do Norte (específico do Reino Unido)
UK_ELECTORAL_ROLL_NUMBER	O Electoral Roll Number (ERN) é o número de identificação emitido para um indivíduo para registro eleitoral no Reino Unido. O formato desse número é especificado pelas Normas do Governo do Reino Unido do Gabinete do Reino Unido.
UK_NATIONAL_HEALTH_SERVICE_NUMBER	O número do Serviço de Saúde Nacional (NHS) é o número exclusivo atribuído a um usuário registrado dos serviços públicos de saúde no Reino Unido.
UK_NATIONAL_INSURANCE_NUMBER	O National Insurance Number (NINO) é um número usado no Reino Unido (Reino Unido) para identificar um indivíduo para o programa

Tipo de dado	Descrição
	de seguro nacional ou o sistema de previdência social. Às vezes o número é referenciado como NI No ou NINO.
UK_PASSPORT_NUMBER	Número do passaporte do Reino Unido (Reino Unido).
UK_UNIQUE_TAXPAYER_REFERENCE_NUMBER	O número de Unique Taxpayer Reference (UTR) do Reino Unido. Um identificador usado pelo governo do Reino Unido para gerenciar o sistema tributário.
UK_VALUE_ADDED_TAX	O VAT é um imposto sobre o consumo que é pago pelo consumidor final. O VAT é pago por cada transação no processo de fabricação e distribuição. Para o Reino Unido, o número do VAT é emitido pelo escritório de VAT da região em que a empresa está estabelecida.
UK_PHONE_NUMBER	Número de telefone do Reino Unido (Reino Unido).

Tipos de dados da Venezuela

Tipo de dado	Descrição
VENEZUELA_DRIVING_LICENSE	O número da carteira de motorista (específico da Venezuela).
VENEZUELA_NATIONAL_IDENTIFICATION_NUMBER	O número de identificação nacional (específico da Venezuela).
VENEZUELA_VALUE_ADDED_TAX	Imposto sobre valor agregado (específico da Venezuela).

Usar a detecção minuciosa de dados confidenciais

> **Note**
>
> As ações minuciosas só estão disponíveis no AWS Glue versões 3.0 e 4.0. Isso inclui a experiência do AWS Glue Studio. As alterações persistentes do log de auditoria também não estão disponíveis na versão 2.0.
> Todas as tarefas visuais do AWS Glue Studio 3.0 e 4.0 terão um script criado que usa automaticamente APIs de ações minuciosas.

A transformação Detectar dados confidenciais permite detectar, mascarar ou remover entidades definidas por você ou que são predefinidas pelo AWS Glue. Além disso, ações minuciosas permitem que você aplique uma ação específica por entidade. Os benefícios adicionais incluem:

- Performance aprimorada à medida que as ações são aplicadas assim que os dados são detectados.
- A opção de incluir ou excluir colunas específicas.
- A capacidade de usar mascaramento parcial. Isso permite mascarar parcialmente as entidades de dados confidenciais detectadas, em vez de mascarar a string inteira. Ambos os parâmetros simples com deslocamentos e regex são aceitos.

A seguir são mostrados trechos de código de APIs de detecção de dados confidenciais e ações minuciosas usados nos exemplos de trabalhos mencionados na próxima seção.

API de detecção: as ações minuciosas usam o novo parâmetro `detectionParameters`:

```
def detect(
    frame: DynamicFrame,
    detectionParameters: JsonOptions,
    outputColumnName: String = "DetectedEntities",
    detectionSensitivity: String = "LOW"
): DynamicFrame = {}
```

Usar APIs de detecção de dados confidenciais com ações minuciosas

As APIs de detecção de dados confidenciais que usam detect analisam os dados fornecidos, determinam se as linhas ou colunas são tipos de entidades de dados confidenciais e executam ações especificadas pelo usuário para cada tipo de entidade.

Usar a API de detecção com ações minuciosas

Use a API detect e especifique `outputColumnName` e `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()
      }
```

O script acima criará um DataFrame com base em um local no Amazon S3 e, em seguida, executará a API `detect`. Como a API `detect` exige que o campo `detectionParameters` (um mapa do nome da entidade para uma lista de todas as configurações de ação a serem usadas para essa

entidade) seja representado pelo objeto `Json0ptions` do AWS Glue, ela também nos permitirá estender a funcionalidade da API.

Para cada ação especificada por entidade, insira uma lista de todos os nomes das colunas às quais deseja aplicar a combinação entidade/ação. Isso permite personalizas as entidades a serem detectadas em cada coluna em seu conjunto de dados e ignorar as entidades que você sabe que não estão em uma coluna específica. Isso também permite que seus trabalhos apresentem uma melhor performance ao não realizar chamadas de detecção desnecessárias para essas entidades e permite a você executar ações exclusivas para cada combinação de coluna e entidade.

Examinando mais de perto o `detectionParameters`, há três tipos de entidades no trabalho de exemplo. São elas: `Phone Number`, `USA_PASSPORT_NUMBER` e `USA_DRIVING_LICENSE`. Para cada um desses tipos de entidade, o AWS Glue executará ações diferentes que são `PARTIAL_REDACT`, `SHA256_HASH`, `REDACT` e `DETECT`. Cada um dos tipos de entidade também tem `sourceColumns` para ser aplicado a ele e/ou `sourceColumnsToExclude`, se detectado.

> **Note**
>
> Somente uma ação de edição no local (`PARTIAL_REDACT`, `SHA256_HASH` ou `REDACT`) pode ser usada por coluna, mas a ação `DETECT` pode ser usada com qualquer uma dessas ações.

O campo `detectionParameters` tem o layout abaixo:

```
ENTITY_NAME -> List[Actions]
{
 "ENTITY_NAME": [{
  Action, // required
  ColumnSpecs,
  ActionOptionsMap
    }],
    "ENTITY_NAME2": [{
  ...
    }]
}
```

Os tipos de `actions` e `actionOptions` estão listados abaixo:

```
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"
    ]
}
```

Depois que o script é executado, os resultados são enviados para a localização específica do Amazon S3. Você pode visualizar seus dados no Amazon S3, mas com os tipos de entidade selecionados sendo sensibilizados com base na ação selecionada. No caso, teríamos uma linha que teria a seguinte aparência:

```
{
```

```
    "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"
}
```

No script acima, o `Phone Number` foi parcialmente redigido com #. O `Passport No` foi transformado em um hash SHA256. O `DL NO#` foi detectado como um número da carteira de motorista dos EUA e foi redigido como "USA_DL", assim como indicado no `detectionParameters`.

> **Note**
>
> A API classifyColumns não está disponível para uso com ações minuciosas devido à natureza da API. Essa API realiza a amostragem de colunas (ajustável pelo usuário, mas tem valores padrão) para realizar a detecção mais rapidamente. Por esse motivo, as ações minuciosas exigem a iteração de cada valor.

Log de auditoria persistente

Um novo recurso introduzido com ações minuciosas (mas também disponível ao usar as APIs normais) é a presença de um log de auditoria persistente. No momento, a execução da API de detecção acrescenta um parâmetro adicional de coluna (o padrão é `DetectedEntities`, mas ele personalizável via `outputColumnName`) com metadados de detecção de PII. Ela agora tem uma chave de metadados "actionUsed", que é uma de `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
        }
    ]
}
```

Até mesmo clientes que usam APIs sem ações minuciosas, como `detect(entityTypesToDetect, outputColumnName)`, verão esse log de auditoria persistente no dataframe resultante.

Os clientes que usam APIs com ações minuciosas verão todas as ações, independentemente de terem sido editadas ou não. Exemplo:

```
+---------------------+---------------
+-------------------------------------------------------------------------------------------------------
+
| 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}]}} |
+--------------------+---------------
+-------------------------------------------------------------------------------------------------------
+
```

Se você não quiser ver a coluna DetectedEntities, basta soltar a coluna adicional em um script personalizado.

API Visual Job do AWS Glue

O AWS Glue fornece uma API que permite aos clientes criar trabalhos de integração de dados usando a API do AWS Glue diretamente de um objeto JSON que representa um fluxo de trabalho de etapa visual. Em seguida, os clientes podem usar o editor visual no AWS Glue Studio para lidar com esses trabalhos.

Para mais informações sobre os tipos de dados da API Visual Job, consulte API Visual Job.

Tópicos

- Design de API e APIs CRUD
- Conceitos básicos
- Limitações de trabalho visual

Design de API e APIs CRUD

Agora as APIs CreateJob e UpdateJob oferecem suporte a um parâmetro opcional adicional, codeGenConfigurationNodes. O fornecimento de uma estrutura JSON não vazia a esse campo resultará no registro do DAG no AWS Glue Studio para o trabalho criado e o código associado que está sendo gerado. Um valor nulo ou string vazia para este campo na criação do trabalho será ignorado.

As atualizações do campo codeGenConfigurationNodes serão feitas por meio da API UpdateJob do AWS Glue de maneira semelhante a CreateJob. Deve-se especificar o campo inteiro em UpdateJob quando o DAG tiver sido alterado conforme desejado. Um valor nulo fornecido será ignorado e nenhuma atualização para o DAG será executada. Uma estrutura ou string vazia fará com que

CodeGenConfigurationNodes seja definido como vazio e qualquer DAG anterior seja removido. A API GetJob retornará um DAG, se houver. A API DeleteJob também excluirá qualquer DAG associado.

Conceitos básicos

Para criar um trabalho, use a ação CreateJob. A entrada da solicitação `CreateJob` terá um campo adicional “codeGenConfigurationNodes”, no qual é possível especificar o objeto DAG no JSON.

Tenha em mente que:

- O campo "codeGenConfigurationNodes" é um mapa de nodeId para o nó.
- Cada nó começa com uma chave que identifica o tipo do nó.
- Só pode haver uma chave especificada, pois um nó só pode ser de um tipo.
- O campo de entrada contém os nós pais do nó atual.

Veja a seguir uma representação de JSON de uma entrada 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"]
        }
      ]
    }
  }
}
```

Atualizar e receber trabalhos

Como UpdateJob também terá um campo “codeGenConfigurationNodes”, o formato de entrada será o mesmo. Consulte a ação UpdateJob.

A ação GetJob também retornará um campo “codeGenConfigurationNodes” no mesmo formato. Consulte a ação GetJob.

Limitações de trabalho visual

Como o parâmetro "codeGenConfigurationNodes" foi adicionado a APIs existentes, quaisquer limitações nessas APIs serão herdadas. Além disso, codeGenConfigurationNodes e alguns nós serão limitados em tamanho. Para mais informações, consulte Estrutura de trabalho.

Programar scripts de ETL

O AWS Glue facilita a gravação e a execução de scripts do Ray. Esta seção descreve os recursos compatíveis do Ray que estão disponíveis no AWS Glue para Ray. Programar scripts do Ray no Python.

Seu script personalizado deve ser compatível com a versão do Ray definida pelo campo `Runtime` na definição do trabalho. Para obter mais informações sobre o `Runtime` na API de trabalhos, consulte the section called “Tarefas”. Para obter mais informações sobre cada ambiente de runtime, consulte the section called “Ambientes de runtime do Ray compatíveis”.

Tópicos

- Tutorial: escrever um script de ETL no AWS Glue para Ray
- Usar o Ray Core e o Ray Data no AWS Glue para Ray
- Fornecer arquivos e bibliotecas do Python para trabalhos do Ray
- Conectar a dados em trabalhos do Ray

Tutorial: escrever um script de ETL no AWS Glue para Ray

O Ray permite escrever e escalar tarefas distribuídas de modo nativo no Python. O AWS Glue para Ray oferece ambientes do Ray sem servidor que você pode acessar tanto em trabalhos quanto em sessões interativas (as sessões interativas do Ray estão em versão prévia). O sistema de trabalhos do AWS Glue fornece uma maneira consistente de gerenciar e executar tarefas, segundo uma agenda, acionadas por um gatilho ou no console do AWS Glue.

A combinação dessas ferramentas do AWS Glue cria um poderoso conjunto de ferramentas que você pode usar para workloads de extração, transformação e carregamento (ETL), um caso de uso popular do AWS Glue. Neste tutorial, você aprenderá o básico sobre como desenvolver essa solução.

Também oferecemos compatibilidade com o uso do AWS Glue for Spark para suas workloads de ETL. Para obter um tutorial sobre escrever scripts do AWS Glue for Spark, consulte the section called “Tutorial: criar um script do Spark”. Para obter mais informações sobre os mecanismos disponíveis, consulte the section called “AWS Glue para Spark e AWS Glue para Ray”. O Ray é capaz de lidar com vários tipos diferentes de tarefas em análise, machine learning (ML) e desenvolvimento de aplicações.

Neste tutorial, você extrairá, transformará e carregará um conjunto de dados CSV hospedado no Amazon Simple Storage Service (Amazon S3). Você começará com o conjunto de dados de registro de viagens da Taxi and Limousine Commission (TLC) da cidade de Nova York, que está armazenado em um bucket público do Amazon S3. Para obter mais informações sobre esse conjunto de dados, consulte Dados abertos na AWS.

Você transformará os dados com transformações predefinidas que estão disponíveis na biblioteca Ray Data. Ray Data é uma biblioteca de preparação de conjuntos de dados projetada pela Ray e incluída por padrão nos ambientes do AWS Glue para Ray. Para obter mais informações sobre as bibliotecas incluídas por padrão, consulte the section called “Módulos fornecidos com trabalhos do Ray”. Em seguida, você gravará os dados transformados em um bucket do Amazon S3 que você controla.

Pré-requisitos: para este tutorial, você precisará de uma conta da AWS com acesso ao AWS Glue e ao Amazon S3.

Etapa 1: criar um bucket no Amazon S3 para armazenar os dados de saída

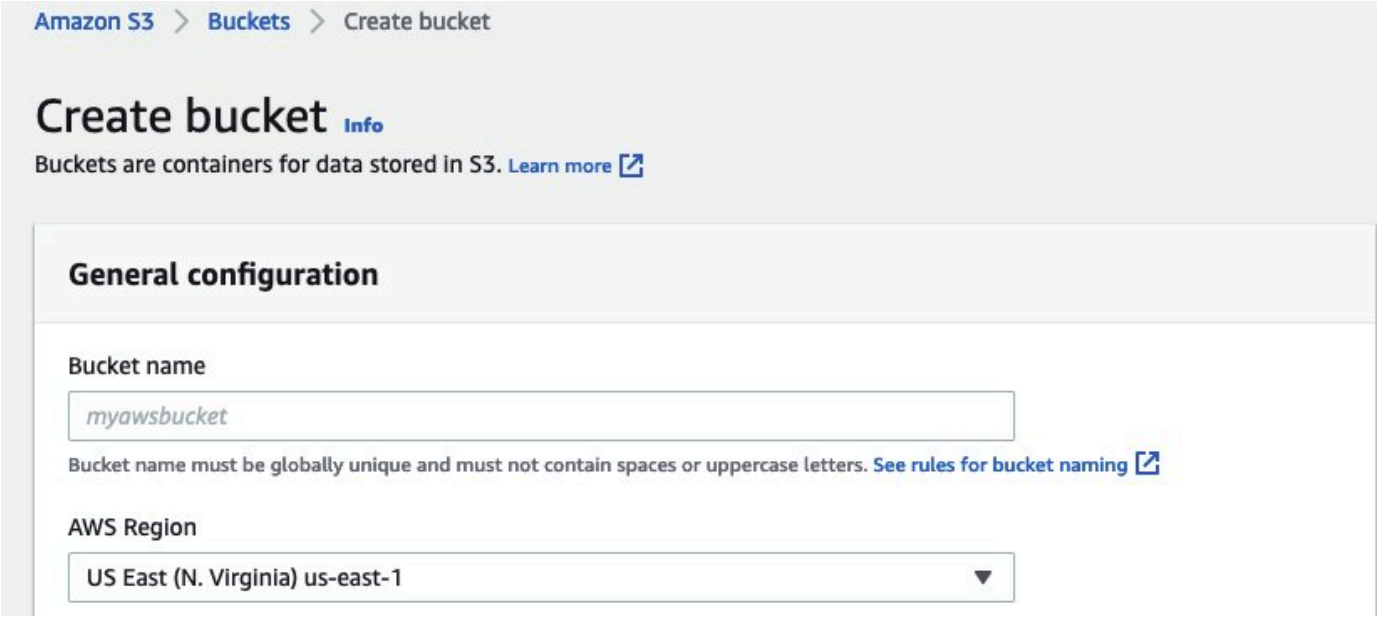
será necessário um bucket do Amazon S3 que você controla para servir como coletor dos dados criados neste tutorial. Você pode criar esse bucket com o procedimento a seguir.

> **Note**
>
> Se quiser gravar os dados em um bucket existente que você controla, pode pular esta etapa. Anote *`yourBucketName`*, o nome do bucket existente, para usar em etapas posteriores.

Para criar um bucket para a saída do trabalho do Ray

- Crie um bucket seguindo as etapas em Criação de um bucket no Guia do usuário do Amazon S3.
  - Ao escolher um nome de bucket, anote *`yourBucketName`*, que você vai referenciar em etapas posteriores.
  - Para outras configurações, as configurações sugeridas fornecidas no console do Amazon S3 devem funcionar bem neste tutorial.

  Por exemplo, a caixa de diálogo de criação do bucket pode ter essa aparência no console do Amazon S3.



Etapa 2: criar uma política e um perfil do IAM para o trabalho do Ray

O trabalho precisará de um perfil do AWS Identity and Access Management (IAM) com o seguinte:

- Permissões concedidas pela política gerenciada pela `AWSGlueServiceRole`. Estas são as permissões básicas necessárias para executar um trabalho do AWS Glue.
- Permissões de nível de acesso de `Read` para o recurso `nyc-tlc/*` do Amazon S3.
- Permissões de nível de acesso de `Write` para o recurso *`yourBucketName`*`/*` do Amazon S3.
- Uma relação de confiança que permita que a entidade principal do `glue.amazonaws.com` assuma o perfil.

Você pode criar esse perfil com o procedimento a seguir.

Para criar um perfil do IAM para o trabalho do AWS Glue para Ray

> **Note**
>
> Você pode criar um perfil do IAM seguindo vários procedimentos diferentes. Para obter mais informações ou opções de como provisionar recursos do IAM, consulte a documentação do AWS Identity and Access Management.

1. Crie uma política que defina as permissões do Amazon S3 descritas anteriormente seguindo as etapas em Criar políticas do IAM (console) com o editor visual no Guia do usuário do IAM.

   - Ao selecionar um serviço, escolha Amazon S3.
   - Ao selecionar permissões para a política, anexe os seguintes conjuntos de ações para os seguintes recursos (mencionados anteriormente):
     - Permissões de nível de acesso de leitura para o recurso `nyc-tlc/*` do Amazon S3.
     - Permissões de nível de acesso de gravação para o recurso *`yourBucketName`*`/*` do Amazon S3.
   - Ao escolher um nome de bucket, anote *`YourPolicyName`*, que você vai referenciar uma etapa posterior.
2. Crie um perfil parra o trabalho do AWS Glue para Ray seguindo as etapas em Criar um perfil para um serviço da AWS (console) no Guia do usuário do IAM.

   - Ao selecionar uma entidade confiável de serviço da AWS, escolha `Glue`. Isso preencherá automaticamente a relação de confiança necessária para o trabalho.
   - Ao selecionar políticas para a política de permissões, anexe as seguintes políticas:
     - `AWSGlueServiceRole`
     - *`YourPolicyName`*
   - Ao selecionar um nome de perfil, anote *`YourRoleName`*, que você vai referenciar em etapas posteriores.

Etapa 3: criar e executar um trabalho do AWS Glue para Ray

Nesta etapa, você cria um trabalho do AWS Glue usando o AWS Management Console, fornece a ele um script de amostra e executa o trabalho. Quando você cria um trabalho, ele cria um local no console para você armazenar, configurar e editar seu script do Ray. Para obter mais informações sobre como criar trabalhos, consulte the section called “Fazer login no console”.

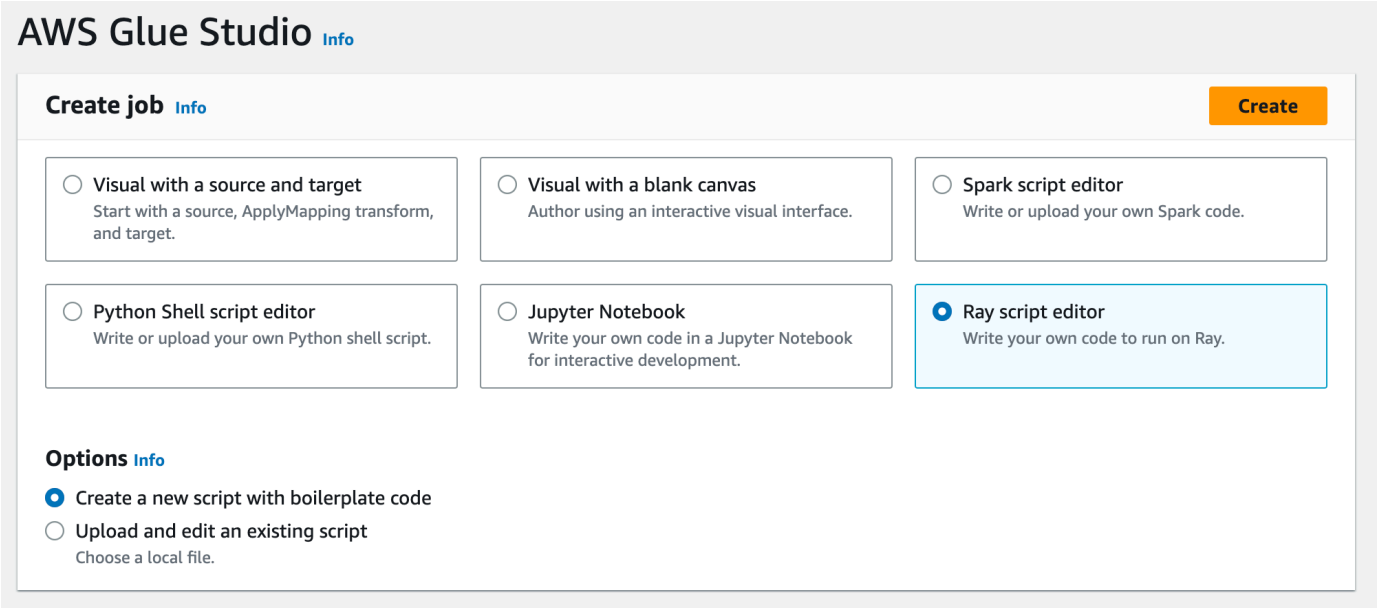
Neste tutorial, abordamos o seguinte cenário de ETL: você deseja ler o conjunto de dados de registro de viagens de janeiro de 2022 da TLC da cidade de Nova York, adicionar uma nova coluna (`tip_rate`) ao conjunto de dados combinando os dados nas colunas existentes, depois remover várias colunas que não são relevantes para sua análise atual e depois deseja gravar os resultados no *`yourBucketName`*. O script Ray a seguir executa essas etapas:

```
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 criar e executar um trabalho do AWS Glue para Ray

1. No AWS Management Console, navegue até a página inicial do AWS Glue.
2. No painel de navegação lateral, escolha Trabalhos de ETL.
3. Em Criar trabalho, escolha Editor de scripts do Ray e depois Criar, como na ilustração a seguir.



4. Cole o texto completo do script no painel Script e substitua qualquer texto existente.
5. Navegue até Detalhes do trabalho e defina a propriedade Perfil do IAM como *YourRoleName*.

6. Escolha Salvar e depois Executar.

Etapa 4: inspecionar a saída

Depois de executar o trabalho do AWS Glue, você deve validar se a saída corresponde às expectativas desse cenário. Para fazer isso, use o seguinte procedimento.

Para validar se o trabalho do Ray foi executado com êxito

1. Na página de detalhes do trabalho, navegue até Execuções.
2. Depois de alguns minutos, você verá uma execução com o Status de execução Bem-sucedida.
3. *Navegue até o console do Amazon S3 em https://console.aws.amazon.com/s3/ e inspecione yourBucketName*. Você deve ver os arquivos gravados no bucket de saída.
4. Leia os arquivos do Parquet e verifique seu conteúdo. Você pode fazer isso com as ferramentas existentes. Se você não tiver um processo para validar arquivos do Parquet, poderá fazer isso no console do AWS Glue com uma sessão interativa do AWS Glue, usando o Spark ou o Ray (em versão prévia).

   Em uma sessão interativa, você tem acesso às bibliotecas Ray Data, Spark ou pandas, que são fornecidas por padrão (com base no mecanismo de sua escolha). Para verificar o conteúdo do arquivo, você pode usar métodos de inspeção comuns que estão disponíveis nessas bibliotecas, métodos como `count`, `schema` e `show`. Para obter mais informações sobre sessões interativas no console, consulte Using notebooks with AWS Glue Studio and AWS Glue.

   Como você confirmou que os arquivos foram gravados no bucket, pode dizer com relativa certeza que, se a saída tiver problemas, eles não serão relacionados à configuração do IAM. Configure a sessão com *yourRoleName* para ter acesso aos arquivos relevantes.

Se você não encontrar os resultados esperados, examine o conteúdo de solução de problemas neste guia para identificar e corrigir a causa do erro. Para interpretar os estados de erro de execução do trabalho, consulte the section called “Status de execução de trabalho”. Você pode encontrar o conteúdo da solução de problemas no capítulo Solução de problemas de AWS Glue. Para erros específicos relacionados aos trabalhos do Ray, consulte the section called “Solucionar problemas de erros do Ray” no capítulo de solução de problemas.

Próximas etapas

Agora você viu e executou um processo de ETL usando o AWS Glue para Ray do início ao fim. Você pode usar os recursos a seguir para entender quais ferramentas o AWS Glue para Ray fornece para transformar e interpretar dados em grande escala.

- Para ter mais informações sobre o modelo de tarefa do Ray, consulte the section called “Usar o Ray Core e o Ray Data no AWS Glue para Ray”. Para ganhar mais experiência no uso de tarefas do Ray, siga os exemplos na documentação do Ray Core. Consulte Ray Core: Ray Tutorials and Examples (2.4.0) na documentação do Ray.
- Para obter orientação sobre as bibliotecas de gerenciamento de dados disponíveis no AWS Glue para Ray, consulte the section called “Conectar-se a dados”. Para obter mais experiência no uso do Ray Data para transformar e gravar conjuntos de dados, siga os exemplos na documentação do Ray Data. Consulte Ray Data: Examples (2.4.0).
- Para obter mais informações sobre configuração e gerenciamento de trabalhos do AWS Glue para Ray, consulte the section called “Trabalhar com trabalhos do Ray”.
- Para obter mais informações sobre escrever scripts do AWS Glue para Ray, continue lendo a documentação nesta seção.

Usar o Ray Core e o Ray Data no AWS Glue para Ray

O Ray é uma estrutura para aumentar verticalmente a escala de scripts do Python distribuindo o trabalho por um cluster. Você pode usar o Ray como uma solução para vários tipos de problemas, então o Ray fornece bibliotecas para otimizar determinadas tarefas. No AWS Glue, nos concentramos em usar o Ray para transformar grandes conjuntos de dados. O AWS Glue é compatível com o Ray Data e partes do Ray Core para facilitar essa tarefa.

O que é o Ray Core?

A primeira etapa da criação de uma aplicação distribuída é identificar e definir o trabalho que pode ser executado simultaneamente. O Ray Core contém as partes do Ray que você usa para definir tarefas que podem ser executadas simultaneamente. O Ray fornece informações de referência e de início rápido que você pode usar para aprender as ferramentas que eles fornecem. Para obter mais informações, consulte What is Ray Core? e Ray Core Quick Start. Para obter mais informações sobre a definição eficaz de tarefas simultâneas no Ray, consulte Tips for first-time users.

> **Tarefas e atores do Ray**
>
> Na documentação do AWS Glue para Ray, podemos nos referir a tarefas e atores, que são conceitos fundamentais do Ray.
> O Ray usa funções e classes do Python como peças de montagem de um sistema de computação distribuído. Como acontece quando as funções e as variáveis do Python se tornam "métodos" e "atributos" quando usadas em uma classe, as funções se tornam "tarefas" e as classes se tornam "atores" quando são usadas no Ray para enviar código aos operadores. Você pode identificar as funções e as classes que podem ser usadas pelo Ray por meio da anotação `@ray.remote`.
> As tarefas e os atores são configuráveis, têm um ciclo de vida e consomem recursos computacionais durante toda a sua existência. Um código que gera erros pode ser rastreado de volta a uma tarefa ou ator quando você descobre a causa raiz dos problemas. Assim, esses termos podem surgir quando você está aprendendo a configurar, monitorar ou depurar trabalhos do AWS Glue para Ray.
> Para começar a aprender como usar tarefas e atores de modo eficaz para criar uma aplicação distribuída, consulte Key Concepts na documentação do Ray.

Ray Core no AWS Glue para Ray

Os ambientes do AWS Glue para Ray gerenciam a formação e a escalação de clusters, bem como a coleta e a visualização de logs. Como gerenciamos essas preocupações, consequentemente limitamos o acesso e a compatibilidade com as APIs no Ray Core que seriam usadas para resolver essas preocupações em um cluster de código aberto.

No ambiente de runtime do `Ray2.4`, não oferecemos compatibilidade com:

- CLI do Ray Core
- CLI do Ray State
- Métodos de utilitários métricos do Prometheus `ray.util.metrics`:
  - Contador
  - Medidor
  - Histograma
- Outras ferramentas de depuração:
  - ray.util.pdb.set_trace

- ray.util.inspect_serializability
- ray.timeline

O que é o Ray Data?

Quando você está se conectando a fontes e destinos de dados, manipulando conjuntos de dados e iniciando transformações comuns, o Ray Data é uma metodologia simples de usar o Ray para resolver problemas de transformação de conjuntos de dados do Ray. Para obter mais informações sobre o uso do Ray Data, consulte Ray Datasets: Distributed Data Preprocessing.

Você pode usar o Ray Data ou outras ferramentas para acessar dados. Para obter mais informações sobre como acessar dados no Ray, consulte the section called “Conectar-se a dados”.

Ray Data no AWS Glue para Ray

O Ray Data é compatível e fornecido por padrão no ambiente de runtime gerenciado do `Ray2.4`. Para obter mais informações sobre os módulos fornecidos, consulte the section called “Módulos fornecidos com trabalhos do Ray”.

Fornecer arquivos e bibliotecas do Python para trabalhos do Ray

Esta seção fornece as informações de que você precisa para usar as bibliotecas do Python com trabalhos do AWS Glue Ray. Você pode usar determinadas bibliotecas comuns incluídas por padrão em todos os trabalhos do Ray. Você também pode fornecer suas próprias bibliotecas do Python para seu trabalho do Ray.

Módulos fornecidos com trabalhos do Ray

Você pode executar fluxos de trabalho de integração de dados em um trabalho do Ray com os seguintes pacotes fornecidos. Esses pacotes estão disponíveis por padrão nos trabalhos do Ray.

AWS Glue version 4.0

No AWS Glue versão 4.0, o ambiente do Ray (runtime do `Ray2.4`) fornece os seguintes pacotes:

- boto3 == 1.26.133
- ray == 2.4.0
- pyarrow == 11.0.0
- pandas== 1.5.3

- numpy == 1.24.3
- fsspec== 2023.4.0

Essa lista inclui todos os pacotes que seriam instalados com o `ray[data] == 2.4.0`. O Ray Data tem compatibilidade imediata.

Fornecer arquivos para o trabalho do Ray

Você pode fornecer arquivos para o trabalho do Ray com o parâmetro `--working-dir`. Forneça a esse parâmetro um caminho para um arquivo.zip hospedado no Amazon S3. Dentro arquivo.zip, os arquivos devem estar contidos em um único diretório de nível superior. Nenhum outro arquivo deve estar no nível superior.

Os arquivos serão distribuídos para cada nó do Ray antes que o script comece a ser executado. Considere como isso pode afetar o espaço em disco disponível para cada nó do Ray. O espaço em disco disponível é determinado pelo WorkerType definido na configuração do trabalho. Se você quiser fornecer os dados do trabalho em grande escala, esse mecanismo não é a solução certa. Para obter mais informações sobre fornecimento de dados no trabalho, consulte the section called “Conectar-se a dados”.

Os arquivos estarão acessíveis como se o diretório tivesse sido fornecido ao Ray por meio do parâmetro `working_dir`. Por exemplo, para ler um arquivo denominado `sample.txt` no diretório de nível superior do arquivo.zip, você pode chamar:

```
@ray.remote
def do_work():
    f = open("sample.txt", "r")
    print(f.read())
```

Para obter mais informações sobre o `working_dir`, consulte a documentação do Ray. Esse atributo se comporta de modo semelhante aos recursos nativos do Ray.

Módulos adicionais do Python para trabalhos do Ray

Módulos adicionais do PyPI

Os trabalhos do Ray usam o pacote de instalação do Python (pip3) para instalar módulos adicionais a serem usados por um script do Ray. Você pode usar o parâmetro `--pip-install` com uma lista

de módulos do Python separados por vírgulas para adicionar um novo módulo ou alterar a versão de um módulo existente.

Por exemplo, para atualizar ou adicionar um novo módulo `scikit-learn`, use o seguinte par chave-valor:

```
"--pip-install", "scikit-learn==0.21.3"
```

Se você tiver módulos personalizados ou patches personalizados, poderá distribuir suas próprias bibliotecas do Amazon S3 com o parâmetro `--s3-py-modules`. Antes de carregar a distribuição, pode ser necessário reempacotá-la e recompilá-la. Siga as diretrizes em the section called “Incluir código do Python em trabalhos do Ray”.

Distribuições personalizadas a partir do Amazon S3

As distribuições personalizadas devem seguir as diretrizes de empacotamento do Ray para dependências. Você pode descobrir como compilar essas distribuições na seção a seguir. Para obter mais informações sobre como o Ray configura dependências, consulte Environment Dependencies (Dependências de ambiente) na documentação do Ray.

Para incluir um distribuível personalizado após avaliar seu conteúdo, carregue o distribuível em um bucket disponível para o perfil do IAM do trabalho. Especifique o caminho do Amazon S3 para um arquivo zip do Python na configuração de parâmetros. Se você estiver fornecendo vários itens distribuíveis, separe-os por vírgulas. Por exemplo:

```
"--s3-py-modules", "s3://s3bucket/pythonPackage.zip"
```

Limitações

Os trabalhos do Ray não são compatíveis com a compilação de código nativo no ambiente de trabalho. Você poderá ser limitado por isso se as dependências do Python dependerem transitivamente do código nativo compilado. Os trabalhos do Ray podem executar os binários fornecidos, mas eles devem ser compilados para Linux no ARM64. Isso significa que você poderá usar o conteúdo as rodas `aarch64manylinux`. Você pode fornecer as dependências nativas em formato compilado reempacotando uma roda de acordo com os padrões do Ray. Normalmente, isso significa remover as pastas `dist-info` para que haja apenas uma pasta na raiz no arquivo compactado.

Você não pode atualizar a versão do `ray` ou do `ray[data]` usando esse parâmetro. Para usar uma nova versão do Ray, você precisará alterar o campo de runtime no trabalho, após lançarmos

a compatibilidade com ele. Para obter mais informações sobre as versões compatíveis do Ray, consulte the section called “Versões do AWS Glue”.

Incluir código do Python em trabalhos do Ray

A Python Software Foundation oferece comportamentos padronizados para empacotar arquivos do Python para uso em diferentes runtimes. O Ray introduz limitações aos padrões de empacotamento que você deve conhecer. O AWS Glue não especifica padrões de empacotamento além dos especificados para o Ray. As instruções a seguir fornecem orientação padrão sobre o empacotamento de pacotes simples do Python.

Empacotar arquivos em um arquivo compactado `.zip`. Um diretório deve estar na raiz do raiz do arquivo compactado. Não deve haver outros arquivos no nível raiz do arquivo compactado, ou isso pode causar um comportamento inesperado. O diretório raiz é o pacote, e o nome dele é usado para se referir ao código do Python ao importá-lo.

Se fornecer um pacote nesse formato para um trabalho do Ray com `--s3-py-modules`, você poderá importar código do Python do pacote no script do Ray.

O pacote pode fornecer um único módulo do Python com alguns arquivos do Python, ou você pode empacotar vários módulos juntos. Ao reempacotar dependências, como bibliotecas do PyPI, verifique se há arquivos e diretórios de metadados ocultos dentro desses pacotes.

> **Warning**
>
> Certos comportamentos do sistema operacional dificultam o cumprimento adequado dessas instruções de empacotamento.
>
> - O OSX pode adicionar arquivos ocultos, como `__MACOSX` no arquivo zip no nível superior.
> - O Windows pode adicionar os arquivos em uma pasta dentro do zip automaticamente, criando involuntariamente uma pasta aninhada.

Os procedimentos a seguir pressupõem que você esteja interagindo com os arquivos no Amazon Linux 2 ou em um sistema operacional similar que forneça uma distribuição do Info-ZIP `zip` e dos utilitários `zipinfo`. Recomendamos o uso dessas ferramentas para evitar comportamentos inesperados.

Para empacotar arquivos do Python para uso no Ray

1. Crie um diretório temporário com o nome do pacote e confirme se o diretório de trabalho é o diretório principal. É possível fazer isso com os seguintes comandos:

```
cd parent_directory
mkdir temp_dir
```

2. Copie os arquivos no diretório temporário e confirme a estrutura de diretórios. O conteúdo desse diretório será acessado diretamente como seu módulo do Python. É possível fazer isso com o seguinte comando da :

```
ls -AR temp_dir
# my_file_1.py
# my_file_2.py
```

3. Compacte a pasta temporária usando zip. É possível fazer isso com os seguintes comandos:

```
zip -r zip_file.zip temp_dir
```

4. Confirme se o arquivo está devidamente empacotado. O *zip_file*`.zip` agora deve ser encontrado em seu diretório de trabalho. É possível inspecioná-lo com o seguinte comando:

```
zipinfo -1 zip_file.zip
# temp_dir/
# temp_dir/my_file_1.py
# temp_dir/my_file_2.py
```

Para reempacotar um pacote do Python para uso no Ray.

1. Crie um diretório temporário com o nome do pacote e confirme se o diretório de trabalho é o diretório principal. É possível fazer isso com os seguintes comandos:

```
cd parent_directory
mkdir temp_dir
```

2. Descompacte o pacote e copie seu conteúdo no diretório temporário. Remova arquivos relacionados ao padrão de empacotamento anterior, deixando somente o conteúdo do módulo. Confirme se a estrutura do arquivo está correta com o seguinte 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. Compacte a pasta temporária usando zip. É possível fazer isso com os seguintes comandos:

```
zip -r zip_file.zip temp_dir
```

4. Confirme se o arquivo está devidamente empacotado. O *zip_file*.zip agora deve ser encontrado em seu diretório de trabalho. É possível inspecioná-lo com o seguinte 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
```

Conectar a dados em trabalhos do Ray

Os trabalhos do Ray do AWS Glue podem usar uma ampla variedade de pacotes do Python criados para você integrar dados rapidamente. Fornecemos um conjunto mínimo de dependências para não encher demais seu ambiente. Para obter mais informações sobre o que está incluído por padrão, consulte the section called “Módulos fornecidos com trabalhos do Ray”.

> **Note**
>
> A extração, transformação e carregamento (ETL) do AWS Glue fornece a abstração DynamicFrame para otimizar fluxos de trabalho de ETL nos quais você resolve as diferenças de esquema entre as linhas do conjunto de dados. O ETL do AWS Glue ETL fornece recursos adicionais: marcadores de trabalhos e agrupamento de arquivos de entrada. No momento, não fornecemos recursos correspondentes em trabalhos do Ray.
> O AWS Glue para Spark permite a conexão direta com determinados formatos de dados, fontes e coletores. No Ray, o SDK da AWS para bibliotecas pandas e bibliotecas atuais de

terceiros atendem substancialmente a essa necessidade. Você precisará consultar essas bibliotecas para entender quais recursos estão disponíveis.

A integração do AWS Glue para Ray com a Amazon VPC não está disponível no momento. Os recursos da Amazon VPC não estarão acessíveis sem uma rota pública. Para obter mais informações sobre o uso do AWS Glue com a Amazon VPC, consulte the section called “Endpoint da VPC (AWS PrivateLink)”.

Bibliotecas comuns para trabalhar com dados no Ray

Ray Data, o Ray Data fornece métodos para lidar com formatos, fontes e coletores de dados comuns. Para obter mais informações sobre formatos e fontes compatíveis com o Ray Data, consulte Entrada/saída na documentação do Ray Data. O Ray Data é uma biblioteca opinativa, não uma biblioteca de uso geral, para lidar com conjuntos de dados.

O Ray fornece algumas orientações sobre casos de uso em que o Ray Data pode ser a melhor solução para seu trabalho. Para obter mais informações, consulte Casos de uso do Ray na documentação do Ray.

AWS SDK for pandas (awswrangler): o AWS SDK for pandas é um produto da AWS que oferece soluções limpas e testadas para leitura e gravação em serviços da AWS quando as transformações gerenciam dados com DataFrames pandas. Para obter mais informações sobre formatos e fontes compatíveis no AWS SDK for pandas, consulte a API Reference na documentação do AWS SDK para pandas.

Para obter exemplos de como ler e gravar dados com o AWS SDK for pandas, consulte Quick Start na documentação do AWS SDK for pandas. O AWS SDK for pandas não fornece transformações para seus dados. Ele só é compatível com leitura e gravação a partir de fontes.

Modin: Modin é uma biblioteca do Python que implementa operações comuns do pandas de uma forma distributível. Para obter mais informações sobre o Modin, consulte a documentação do Modin. O próprio Modin não é compatível com leitura e gravação a partir de fontes. Ele fornece implementações distribuídas de transformações comuns. O Modin é compatível com o AWS SDK for pandas.

Ao executar o Modin e o AWS SDK for pandas juntos em um ambiente do Ray, você pode realizar tarefas comuns de ETL com resultados de alta performance. Para obter mais informações sobre como usar o Modin com o AWS SDK para pandas, consulte At scale no AWS SDK for pandas.

Outras estruturas: para obter mais informações sobre estruturas compatíveis com o Ray, consulte O ecossistema do Ray na documentação do Ray. Não há compatibilidade com outras estruturas no AWS Glue para Ray.

Conectar-se a dados por meio do catálogo de dados

O gerenciamento dados por meio do catálogo de dados em conjunto com os trabalhos do Ray é compatível com SDK da AWSpara pandas. Para obter mais informações, consulte Glue Catalog no site do AWS SDK for pandas.

Usando esse serviço com um AWS SDK

AWS kits de desenvolvimento de software (SDKs) estão disponíveis para muitas linguagens de programação populares. Cada SDK fornece uma API, exemplos de código e documentação que facilitam a criação de aplicações em seu idioma preferido pelos desenvolvedores.

Documentação do SDK	Exemplos de código
AWS SDK for C++	AWS SDK for C++ exemplos de código
AWS CLI	AWS CLI exemplos de código
AWS SDK for Go	AWS SDK for Go exemplos de código
AWS SDK for Java	AWS SDK for Java exemplos de código
AWS SDK for JavaScript	AWS SDK for JavaScript exemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin exemplos de código
AWS SDK for .NET	AWS SDK for .NET exemplos de código
AWS SDK for PHP	AWS SDK for PHP exemplos de código
AWS Tools for PowerShell	Ferramentas para exemplos PowerShell de código
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) exemplos de código
AWS SDK for Ruby	AWS SDK for Ruby exemplos de código
AWS SDK para Rust	AWS SDK para Rust exemplos de código
SDK da AWS para SAP ABAP	SDK da AWS para SAP ABAP exemplos de código
AWS SDK for Swift	AWS SDK for Swift exemplos de código

Para obter exemplos específicos deste serviço, consulte AWS Glue Exemplos de código de API usando AWS SDKs.

> **Exemplo de disponibilidade**
>
> Você não consegue encontrar o que precisa? Solicite um código de exemplo no link Fornecer feedback na parte inferior desta página.

AWS Glue API

Esta seção descreve os tipos de dados e primitivos usados pelos AWS Glue SDKs e pelas ferramentas. Há três maneiras gerais de interagir AWS Glue programaticamente fora do AWS Management Console, cada uma com sua própria documentação:

- As bibliotecas de SDKs de linguagens permitem que você acesse recursos do AWS nas linguagens de programação comuns. Encontre mais informações em Ferramentas para criar na AWS.
- O AWS CLI permite que você acesse AWS recursos a partir da linha de comando. Encontre mais informações em Referência de comandos da AWS CLI.
- AWS CloudFormation permite que você defina um conjunto de AWS recursos a serem provisionados juntos de forma consistente. Encontre mais informações em AWS CloudFormation: referência do tipo de AWS Glue recurso.

Esta seção documenta os primitivos compartilhados independentemente desses SDKs e ferramentas. As ferramentas usam a Referência de API AWS Glue da Web para se comunicar com AWS.

Sumário

- StorageDescriptor estrutura
- SchemaReference estrutura
- SerDeInfo estrutura
- Estrutura Order
- SkewedInfo estrutura
- TableVersion estrutura
- TableError estrutura
- TableVersionError estrutura
- SortCriterion estrutura
- TableIdentifier estrutura
- KeySchemaElement estrutura
- PartitionIndex estrutura
- PartitionIndexDescriptor estrutura
- BackfillError estrutura
- IcebergInput estrutura
- OpenTableFormatInput estrutura
- ViewDefinition estrutura
- ViewDefinitionInput estrutura
- ViewRepresentation estrutura
- ViewRepresentationInput estrutura
- Operações
- CreateTable ação (Python: create_table)
- UpdateTable ação (Python: update_table)
- DeleteTable ação (Python: delete_table)
- BatchDeleteTable ação (Python: batch_delete_table)
- GetTable ação (Python: get_table)
- GetTables ação (Python: get_tables)
- GetTableVersion ação (Python: get_table_version)
- GetTableVersions ação (Python: get_table_versions)
- DeleteTableVersion ação (Python: delete_table_version)

- CodeGenConfigurationNode estrutura
- Estrutura do JDBC ConnectorOptions
- StreamingDataPreviewOptions estrutura
- AthenaConnectorSource estrutura
- Estrutura do JDBC ConnectorSource
- SparkConnectorSource estrutura
- CatalogSource estrutura
- Estrutura do MySQL CatalogSource
- Estrutura do PostgreSQL CatalogSource
- Estrutura do OracleSQL CatalogSource
- Estrutura Microsoft SQL ServerCatalogSource
- CatalogKinesisSource estrutura
- DirectKinesisSource estrutura
- KinesisStreamingSourceOptions estrutura
- CatalogKafkaSource estrutura
- DirectKafkaSource estrutura
- KafkaStreamingSourceOptions estrutura
- RedshiftSource estrutura
- AmazonRedshiftSource estrutura
- AmazonRedshiftNodeData estrutura
- AmazonRedshiftAdvancedOption estrutura
- Estrutura Option
- Estrutura S3 CatalogSource
- Estrutura S3 SourceAdditionalOptions
- Estrutura S3 CsvSource
- Estrutura DirectJDBCSource
- Estrutura S3 DirectSourceAdditionalOptions
- Estrutura S3 JsonSource
- Estrutura S3 ParquetSource
- Estrutura S3 DeltaSource

- Estrutura S3 CatalogDeltaSource
- CatalogDeltaSource estrutura
- Estrutura S3 HudiSource
- Estrutura S3 CatalogHudiSource
- CatalogHudiSource estrutura
- Estrutura do DynamoDB CatalogSource
- RelationalCatalogSource estrutura
- Estrutura do JDBC ConnectorTarget
- SparkConnectorTarget estrutura
- BasicCatalogTarget estrutura
- Estrutura do MySQL CatalogTarget
- Estrutura do PostgreSQL CatalogTarget
- Estrutura do OracleSQL CatalogTarget
- Estrutura Microsoft SQL ServerCatalogTarget
- RedshiftTarget estrutura
- AmazonRedshiftTarget estrutura
- UpsertRedshiftTargetOptions estrutura
- Estrutura S3 CatalogTarget
- Estrutura S3 GlueParquetTarget
- CatalogSchemaChangePolicy estrutura
- Estrutura S3 DirectTarget
- Estrutura S3 HudiCatalogTarget
- Estrutura S3 HudiDirectTarget
- Estrutura S3 DeltaCatalogTarget
- Estrutura S3 DeltaDirectTarget
- DirectSchemaChangePolicy estrutura
- ApplyMapping estrutura
- Estrutura Mapping
- SelectFields estrutura
- DropFields estrutura

- RenameField estrutura
- Estrutura Spigot
- Estrutura Join
- JoinColumn estrutura
- SplitFields estrutura
- SelectFromCollection estrutura
- FillMissingValues estrutura
- Estrutura Filter
- FilterExpression estrutura
- FilterValue estrutura
- CustomCode estrutura
- Estrutura SparkSQL
- SqlAlias estrutura
- DropNullFields estrutura
- NullCheckBoxList estrutura
- NullValueField estrutura
- Estrutura Datatype
- Estrutura Merge
- Estrutura Union
- Estrutura PIIDetection
- Estrutura Aggregate
- DropDuplicates estrutura
- GovernedCatalogTarget estrutura
- GovernedCatalogSource estrutura
- AggregateOperation estrutura
- GlueSchema estrutura
- GlueStudioSchemaColumn estrutura
- GlueStudioColumn estrutura
- DynamicTransform estrutura
- TransformConfigParameter estrutura

- Tipos de dados
- JobRun estrutura
- Estrutura Predecessor
- JobBookmarkEntry estrutura
- BatchStopJobRunSuccessfulSubmission estrutura
- BatchStopJobRunError estrutura
- NotificationProperty estrutura
- Operações
- StartJobRun ação (Python: start_job_run)
- BatchStopJobRun ação (Python: batch_stop_job_run)
- GetJobRun ação (Python: get_job_run)
- GetJobRuns ação (Python: get_job_runs)
- GetJobBookmark ação (Python: get_job_bookmark)
- GetJobBookmarks ação (Python: get_job_bookmarks)
- ResetJobBookmark ação (Python: reset_job_bookmark)

- Acionadores
  - Tipos de dados
  - Estrutura de acionador
  - Estrutura TriggerUpdate
  - Estrutura Predicate
  - Estrutura Condition
  - Estrutura Action
  - Estrutura EventBatchingCondition
  - Operações
  - Ação CreateTrigger (Python: create_trigger)
  - Ação StartTrigger (Python: start_trigger)
  - Ação GetTrigger (Python: get_trigger)
  - Ação GetTriggers (Python: get_triggers)
  - Ação UpdateTrigger (Python: update_trigger)
  - Ação StopTrigger (Python: stop_trigger)

- Estrutura de GlueEncryptionException
- Estrutura IdempotentParameterMismatchException
- Estrutura IllegalWorkflowStateException
- Estrutura InternalServiceException
- Estrutura InvalidExecutionEngineException
- Estrutura InvalidInputException
- Estrutura InvalidStateException
- Estrutura InvalidTaskStatusTransitionException
- Estrutura JobDefinitionErrorException
- Estrutura JobRunInTerminalStateException
- Estrutura JobRunInvalidStateTransitionException
- Estrutura JobRunNotInTerminalStateException
- Estrutura LateRunnerException
- Estrutura NoScheduleException
- Estrutura OperationTimeoutException
- Estrutura ResourceNotReadyException
- Estrutura ResourceNumberLimitExceededException
- Estrutura SchedulerNotRunningException
- Estrutura SchedulerRunningException
- Estrutura SchedulerTransitioningException
- Estrutura UnrecognizedRunnerException
- Estrutura ValidationException
- Estrutura VersionMismatchException

APIs de segurança no AWS Glue

A API de segurança descreve os tipos de dados de segurança e a API relacionada à segurança no AWS Glue.

Tipos de dados

- Estrutura DataCatalogEncryptionSettings

- Estrutura EncryptionAtRest
- Estrutura ConnectionPasswordEncryption
- Estrutura EncryptionConfiguration
- Estrutura S3Encryption
- Estrutura CloudWatchEncryption
- Estrutura JobBookmarksEncryption
- Estrutura SecurityConfiguration
- Estrutura GluePolicy

Estrutura DataCatalogEncryptionSettings

Contém informações de configuração para manter a segurança do Catálogo de dados.

Campos

- `EncryptionAtRest` – Um objeto EncryptionAtRest.

  Especifica a configuração de criptografia em repouso para o Catálogo de dados.
- `ConnectionPasswordEncryption` – Um objeto ConnectionPasswordEncryption.

  Quando a proteção por senha de conexão é habilitada, o Catálogo de dados usa uma chave fornecida pelo cliente para criptografar a senha como parte do `CreateConnection` ou `UpdateConnection` armazená-la no `ENCRYPTED_PASSWORD` campo nas propriedades da conexão. Você pode habilitar a criptografia de catálogo ou apenas a criptografia de senha.

Estrutura EncryptionAtRest

Especifica a configuração de criptografia em repouso para o Catálogo de dados.

Campos

- `CatalogEncryptionMode` – Obrigatório: string UTF-8 (valores válidos: `DISABLED` | `SSE-KMS="SSEKMS"` | `SSE-KMS-WITH-SERVICE-ROLE="SSEKMSWITHSERVICEROLE"`).

  O modo de criptografia em repouso para a configuração de dados do Catálogo de dados.
- `SseAwsKmsKeyId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O ID da chave do AWS KMS a ser usada para a criptografia em repouso.

- `CatalogEncryptionServiceRole` – String UTF-8 correspondente a Custom string pattern #24.

  O perfil assumido pelo AWS Glue para criptografar e descriptografar os objetos do Catálogo de Dados em nome do chamador.

Estrutura ConnectionPasswordEncryption

A estrutura de dados usada pelo Catálogo de dados para criptografar a senha como parte do `CreateConnection` ou `UpdateConnection` e armazená-la no `ENCRYPTED_PASSWORD` campo nas propriedades da conexão. Você pode habilitar a criptografia de catálogo ou apenas a criptografia de senha.

Quando uma solicitação `CreationConnection` chega contendo uma senha, o Data Catalog primeiro criptografa a senha usando a sua chave do AWS KMS. Em seguida, criptografa todo o objeto de conexão novamente se a criptografia do catálogo também estiver habilitada.

Essa criptografia requer que as permissões de chave do AWS KMS sejam definidas para habilitar ou restringir o acesso à chave da senha de acordo com os seus requisitos de segurança. Por exemplo, você pode querer que apenas administradores tenham permissão para descriptografar a chave da senha.

Campos

- `ReturnConnectionPasswordEncrypted`: obrigatório: booleano.

  Quando o `ReturnConnectionPasswordEncrypted` sinalizador estiver definido como "verdadeiro", as senhas permanecem criptografadas nas respostas de `GetConnection` e `GetConnections`. Esta criptografia entra em vigor independentemente da criptografia de catálogo.

- `AwsKmsKeyId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Uma chave do AWS KMS usada para criptografar a senha de conexão.

  Se a proteção por senha da conexão estiver habilitada, o autor da chamada de `CreateConnection` e `UpdateConnection` precisará, pelo menos, da permissão `kms:Encrypt`

na chave do AWS KMS especificada para criptografar as senhas antes de armazená-las no Data Catalog.

Você pode definir a permissão de descriptografia para habilitar ou restringir o acesso à chave de senha, de acordo com os seus requisitos de segurança.

Estrutura EncryptionConfiguration

Especifica uma configuração de criptografia.

Campos

- `S3Encryption` – Uma matriz de objetos S3Encryption.

  A configuração da criptografia para os dados do Amazon Simple Storage Service (Amazon S3).
- `CloudWatchEncryption` – Um objeto CloudWatchEncryption.

  A configuração da criptografia para o Amazon CloudWatch.
- `JobBookmarksEncryption` – Um objeto JobBookmarksEncryption.

  A configuração da criptografia para marcadores de trabalho.

Estrutura S3Encryption

Especifica como os dados do Amazon Simple Storage Service (Amazon S3) devem ser criptografados.

Campos

- `S3EncryptionMode` – String UTF-8 (valores válidos: `DISABLED` | `SSE-KMS="SSEKMS"` | `SSE-S3="SSES3"`).

  O modo de criptografia a ser usado para dados do Amazon S3.
- `KmsKeyArn` – String UTF-8 correspondente a Custom string pattern #25.

  O nome de recurso da Amazon (ARN) da chave do KMS a ser usada para criptografar os dados.

Estrutura CloudWatchEncryption

Especifica como os dados do Amazon CloudWatch devem ser criptografados.

Campos

- `CloudWatchEncryptionMode`: string UTF-8 (valores válidos: `DISABLED` | `SSE-KMS="SSEKMS"`).

  O modo de criptografia a ser usado para dados do CloudWatch.
- `KmsKeyArn` – String UTF-8 correspondente a Custom string pattern #25.

  O nome de recurso da Amazon (ARN) da chave do KMS a ser usada para criptografar os dados.

Estrutura JobBookmarksEncryption

Especifica como os dados de marcadores de trabalho devem ser criptografados.

Campos

- `JobBookmarksEncryptionMode`: string UTF-8 (valores válidos: `DISABLED` | `CSE-KMS="CSEKMS"`).

  O modo de criptografia a ser usado para dados de marcadores de trabalho.
- `KmsKeyArn` – String UTF-8 correspondente a Custom string pattern #25.

  O nome de recurso da Amazon (ARN) da chave do KMS a ser usada para criptografar os dados.

Estrutura SecurityConfiguration

Especifica uma configuração de segurança.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da configuração de segurança.
- `CreatedTimeStamp` – Timestamp.

O momento em que esta configuração de segurança foi criada.

- `EncryptionConfiguration` – Um objeto EncryptionConfiguration.

  A configuração de criptografia associada a essa configuração de segurança.

Estrutura GluePolicy

Uma estrutura para retornar uma política de recursos.

Campos

- `PolicyInJson` – String UTF-8, com pelo menos 2 bytes de comprimento.

  Contém o documento da política solicitada no formato JSON.
- `PolicyHash` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Contém o valor de hash associado a essa política.
- `CreateTime` – Timestamp.

  A data e hora em que a política foi criada.
- `UpdateTime` – Timestamp.

  A data e hora em que a política foi atualizada pela última vez.

Operações

- Ação GetDataCatalogEncryptionSettings (Python: get_data_catalog_encryption_settings)
- Ação PutDataCatalogEncryptionSettings (Python: put_data_catalog_encryption_settings)
- Ação PutResourcePolicy (Python: put_resource_policy)
- Ação GetResourcePolicy (Python: get_resource_policy)
- Ação DeleteResourcePolicy (Python: delete_resource_policy)
- Ação CreateSecurityConfiguration (Python: create_security_configuration)
- Ação DeleteSecurityConfiguration (Python: delete_security_configuration)
- Ação GetSecurityConfiguration (Python: get_security_configuration)

- Ação GetSecurityConfigurations (Python: get_security_configurations)
- Ação GetResourcePolicies (Python: get_resource_policies)

Ação GetDataCatalogEncryptionSettings (Python: get_data_catalog_encryption_settings)

Recupera a configuração de segurança de um determinado catálogo.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do Catálogo de dados para o qual recuperar a configuração de segurança. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

Resposta

- `DataCatalogEncryptionSettings` – Um objeto DataCatalogEncryptionSettings.

  A configuração de segurança solicitada.

Erros

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Ação PutDataCatalogEncryptionSettings (Python: put_data_catalog_encryption_settings)

Define a configuração de segurança de um determinado catálogo. Depois que a configuração tiver sido definida, a criptografia especificada será aplicada a cada gravação de catálogo a partir de então.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do Catálogo de dados para o qual definir a configuração de segurança. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.
- `DataCatalogEncryptionSettings` – Obrigatório: um objeto DataCatalogEncryptionSettings.

  A configuração de segurança a ser definida.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Ação PutResourcePolicy (Python: put_resource_policy)

Define a política de recurso de catálogo de dados para controle de acesso.

Solicitação

- `PolicyInJson` – Obrigatório: String UTF-8, com pelo menos 2 bytes de comprimento.

  Contém o documento de política a ser definido, no formato JSON.
- `ResourceArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  Não use. Apenas para uso interno.
- `PolicyHashCondition` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O valor de hash retornado quando a política anterior foi definida usando `PutResourcePolicy`. O objetivo é impedir modificações simultâneas de uma política. Não use esse parâmetro se nenhuma política anterior tiver sido definida.

- `PolicyExistsCondition` – String UTF-8 (valores válidos: `MUST_EXIST` | `NOT_EXIST` | `NONE`).

  Um valor de `MUST_EXIST` é usado para atualizar uma política. Um valor de `NOT_EXIST` é usado para criar uma nova política. Se um valor de `NONE` ou um valor nulo for usado, a chamada não dependerá da existência de uma política.

- `EnableHybrid`: string UTF-8 (valores válidos: `TRUE` | `FALSE`).

  Se for `'TRUE'`, indica que você está usando os dois métodos para conceder acesso entre contas aos recursos do Data Catalog:

  - Atualizando diretamente a política de recursos com `PutResourcePolicy`
  - Usando a opção Grant permissions (Conceder permissões) no AWS Management Console.

  Deve ser definida como `'TRUE'` se você já tiver usado o Console de Gerenciamento para conceder acesso entre contas, caso contrário, a chamada falhará. O padrão é “FALSE” (FALSO).

Resposta

- `PolicyHash` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um hash da política que acabou de ser definida. Isso deve ser incluído em uma chamada subsequente que substitui ou atualiza essa política.

Erros

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ConditionCheckFailureException`

Ação GetResourcePolicy (Python: get_resource_policy)

Recupera uma política de recurso especificada.

Solicitação

- `ResourceArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O ARN do recurso do AWS Glue para o qual se deseja recuperar a política de recurso. Se não for fornecido, a política de recurso do Data Catalog será retornada. Use `GetResourcePolicies` para exibir todas as políticas de recursos existentes. Para obter mais informações, consulte Especificar ARNs de recurso do AWS Glue.

Resposta

- `PolicyInJson` – String UTF-8, com pelo menos 2 bytes de comprimento.

  Contém o documento da política solicitada no formato JSON.
- `PolicyHash` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Contém o valor de hash associado a essa política.
- `CreateTime` – Timestamp.

  A data e hora em que a política foi criada.
- `UpdateTime` – Timestamp.

  A data e hora em que a política foi atualizada pela última vez.

Erros

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Ação DeleteResourcePolicy (Python: delete_resource_policy)

Exclui uma política especificada.

Solicitação

- `PolicyHashCondition` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O valor de hash retornado quando essa política foi definida.

- `ResourceArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O ARN do recurso do AWS Glue para a política de recursos a ser excluída.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ConditionCheckFailureException`

Ação CreateSecurityConfiguration (Python: create_security_configuration)

Cria uma nova configuração de segurança; Uma configuração de segurança é um conjunto de propriedades de segurança que pode ser usado pelo AWS Glue. Você pode usar uma configuração de segurança para criptografar dados em repouso. Para obter informações sobre como usar as configurações de segurança no AWS Glue, consulte Criptografar dados gravados por crawlers, trabalhos e endpoints de desenvolvimento.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da nova configuração de segurança.
- `EncryptionConfiguration` – Obrigatório: um objeto EncryptionConfiguration.

  A configuração de criptografia da nova configuração de segurança.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome atribuído à nova configuração de segurança.
- `CreatedTimestamp` – Timestamp.

  O momento em que a nova configuração de segurança foi criada.

Erros

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

Ação DeleteSecurityConfiguration (Python: delete_security_configuration)

Exclui uma configuração de segurança especificada.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da configuração de segurança a ser excluída.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetSecurityConfiguration (Python: get_security_configuration)

Recupera uma configuração de segurança especificada.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da configuração de segurança a ser recuperada.

Resposta

- `SecurityConfiguration` – Um objeto SecurityConfiguration.

  A configuração de segurança solicitada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetSecurityConfigurations (Python: get_security_configurations)

Recupera uma lista de todas as configurações de segurança.

Solicitação

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de resultados a serem retornados.
- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `SecurityConfigurations` – Uma matriz de objetos SecurityConfiguration.

  Uma lista de configurações de segurança.
- `NextToken` – String UTF-8.

  Um token de continuação, se houver mais configurações de segurança para retornar.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetResourcePolicies (Python: get_resource_policies)

Recupera as políticas de recursos definidas em recursos individuais pelo AWS Resource Access Manager durante concessões de permissões entre contas. Recupera também a política de recurso do Data Catalog.

Se você tiver ativado a criptografia de metadados nas definições do Data Catalog e não tiver permissão na chave do AWS KMS, a operação não poderá retornar a política de recurso do Data Catalog.

Solicitação

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma solicitação de continuação.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo de uma lista a ser retornada.

Resposta

- `GetResourcePoliciesResponseList` – Uma matriz de objetos GluePolicy.

  Uma lista das políticas de recursos individuais e a política de recursos de nível de conta.
- `NextToken` – String UTF-8.

  Um token de continuação, se a lista retornada não contiver a política de recurso mais recente disponível.

Erros

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

API do Catalog

A API do catálogo descreve os tipos de dados e a API relacionada ao trabalho com os catálogos no AWS Glue.

Tópicos

- API de banco de dados
- API de tabela
- API de partição
- API de conexão

- API User-defined Function
- Importar um catálogo do Athena para AWS Glue

API de banco de dados

O Banco de dados da API descreve os tipos de dados do banco de dados e inclui a API para criar, excluir, localizar, atualizar e listar bancos de dados.

Tipos de dados

- Estrutura Database
- Estrutura DatabaseInput
- Estrutura PrincipalPermissions
- Estrutura DataLakePrincipal
- Estrutura DatabaseIdentifier
- Estrutura FederatedDatabase

Estrutura Database

O objeto `Database` representa um agrupamento lógico de tabelas que podem residir em uma metastore do Hive ou um RDBMS.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados. Para a compatibilidade com o Hive, as letras são transformadas em minúsculas quando a tabela é armazenada.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do banco de dados.
- `LocationUri` – URI (Uniform Resource Identifier), maior que 1 ou maior que 1024 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A localização do banco de dados (por exemplo, um caminho de HDFS).

- `Parameters` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

  Esses pares de chave/valor definem parâmetros e propriedades do banco de dados.
- `CreateTime` – Timestamp.

  A hora em que o banco de dados de metadados foi criado no catálogo.
- `CreateTableDefaultPermissions` – Uma matriz de objetos Permissões da entidade principal.

  Cria um conjunto de permissões padrão na tabela para as entidades principais. Usado pelo AWS Lake Formation. Não usado no curso normal de operações do AWS Glue.
- `TargetDatabase` – Um objeto DatabaseIdentifier.

  Uma estrutura de `DatabaseIdentifier` que descreve um banco de dados de destino para vinculação de recursos.
- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que o banco de dados reside.
- `FederatedDatabase` – Um objeto FederatedDatabase.

  Uma estrutura `FederatedDatabase` que referencia uma entidade fora do AWS Glue Data Catalog.

Estrutura DatabaseInput

A estrutura usada para criar ou atualizar um banco de dados.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados. Para a compatibilidade com o Hive, as letras são transformadas em minúsculas quando a tabela é armazenada.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do banco de dados.

- `LocationUri` – URI (Uniform Resource Identifier), maior que 1 ou maior que 1024 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A localização do banco de dados (por exemplo, um caminho de HDFS).

- `Parameters` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

  Esses pares de chave/valor definem parâmetros e propriedades do banco de dados.

  Esses pares de chave/valor definem parâmetros e propriedades do banco de dados.

- `CreateTableDefaultPermissions` – Uma matriz de objetos Permissões da entidade principal.

  Cria um conjunto de permissões padrão na tabela para as entidades principais. Usado pelo AWS Lake Formation. Não usado no curso normal de operações do AWS Glue.

- `TargetDatabase` – Um objeto DatabaseIdentifier.

  Uma estrutura de `DatabaseIdentifier` que descreve um banco de dados de destino para vinculação de recursos.

- `FederatedDatabase` – Um objeto FederatedDatabase.

  Uma estrutura `FederatedDatabase` que referencia uma entidade fora do AWS Glue Data Catalog.

Estrutura PrincipalPermissions

Permissões concedidas a uma entidade principal.

Campos

- `Principal` – Um objeto Entidade principal do Data Lake.

A entidade principal que recebe permissões.

- `Permissions` – Uma matriz de strings UTF-8.

  As permissões que são concedidas à entidade principal.

Estrutura DataLakePrincipal

A entidade principal do AWS Lake Formation.

Campos

- `DataLakePrincipalIdentifier`: string UTF-8, não menos do que 1 ou superior a 255 bytes de comprimento.

  Um identificador para a entidade principal do AWS Lake Formation.

Estrutura DatabaseIdentifier

Uma estrutura de que descreve um banco de dados de destino para vinculação de recursos.

Campos

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que o banco de dados reside.
- `DatabaseName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados do catálogo.
- `Region` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Região do banco de dados de destino.

Estrutura FederatedDatabase

Um banco de dados que aponta para uma entidade fora do AWS Glue Data Catalog.

Campos

- `Identifier` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Single-line string pattern.

  Um identificador exclusivo do banco de dados federado.
- `ConnectionName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da conexão com o metastore externo.

Operações

- Ação CreateDatabase (Python: create_database)
- Ação UpdateDatabase (Python: update_database)
- Ação DeleteDatabase (Python: delete_database)
- Ação GetDatabase (Python: get_database)
- Ação GetDatabases (Python: get_databases)

Ação CreateDatabase (Python: create_database)

Cria um novo banco de dados em um catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que o banco de dados será criado. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.
- `DatabaseInput` – Obrigatório: um objeto DatabaseInput.

  Os metadados para o banco de dados.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

As tags que você atribui ao banco de dados.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`
- `FederatedResourceAlreadyExistsException`

Ação UpdateDatabase (Python: update_database)

Atualiza uma definição de banco de dados existente no catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que o banco de dados de metadados reside. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados a ser atualizado no catálogo. Para a compatibilidade com o Hive, as letras são transformadas em minúsculas.

- `DatabaseInput` – Obrigatório: um objeto DatabaseInput.

Um objeto `DatabaseInput` que especifica a nova definição do banco de dados de metadados no catálogo.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`

Ação DeleteDatabase (Python: delete_database)

Remove um banco de dados especificado de um Catálogo de dados.

> **Note**
>
> Depois de concluir essa operação, você não terá mais acesso às tabelas (e a todas as versões de tabela e partições que podem pertencer às tabelas) e às funções definidas pelo usuário no banco de dados excluído. O AWS Glue exclui esses recursos "órfãos" de forma assíncrona e pontual, a critério do serviço.
> Para garantir exclusão imediata de todos os recursos relacionados, antes de chamar `DeleteDatabase`, use `DeleteTableVersion` ou `BatchDeleteTableVersion`, `DeletePartition` ou `BatchDeletePartition`, `DeleteUserDefinedFunction`, e `DeleteTable` ou `BatchDeleteTable`, para excluir qualquer recurso que pertence à tabela.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que o banco de dados reside. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.
- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados a ser excluído. Para a compatibilidade com o Hive, ele deve ser todo inserido em letras minúsculas.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Ação GetDatabase (Python: get_database)

Recupera a definição de um banco de dados especificado.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que o banco de dados reside. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados a ser recuperado. Para a compatibilidade com o Hive, ele deve ser inserido todo em letras minúsculas.

Resposta

- `Database` – Um objeto Banco de dados.

  A definição do banco de dados especificado no Catálogo de dados.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `FederationSourceException`

Ação GetDatabases (Python: get_databases)

Recupera todos os bancos de dados definidos em determinado Catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados de onde `Databases` será recuperado. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

- `MaxResults`: número (inteiro), não menos do que 1 ou superior a 100.

O número máximo de bancos de dados a ser retornado em uma resposta.

- `ResourceShareType` – String UTF-8 (valores válidos: `FOREIGN` | `ALL` | `FEDERATED`).

  Permite que você especifique que deseja listar os bancos de dados compartilhados com sua conta. Os valores permitidos são `FEDERATED`, `FOREIGN` ou `ALL`.

  - Se definido como `FEDERATED`, listará os bancos de dados federados (que referenciam uma entidade externa) compartilhados com sua conta.
  - Se definido como `FOREIGN`, listará os bancos de dados compartilhados com sua conta.
  - Se definido como `ALL`, listará os bancos de dados compartilhados com sua conta, bem como os bancos de dados em sua conta local.

Resposta

- `DatabaseList` – Obrigatório: uma matriz de objetos Banco de dados.

  Uma lista de objetos `Database` do catálogo especificado.

- `NextToken` – String UTF-8.

  Um token de continuação para paginação da lista de tokens retornada, retornado se o segmento atual da lista não for o último.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

API de tabela

A Tabela da API descreve os tipos de dados e operações associadas a tabelas.

Tipos de dados

- Estrutura Table

- TableInput estrutura
- FederatedTable estrutura
- Estrutura da coluna
- StorageDescriptor estrutura
- SchemaReference estrutura
- SerDeInfo estrutura
- Estrutura Order
- SkewedInfo estrutura
- TableVersion estrutura
- TableError estrutura
- TableVersionError estrutura
- SortCriterion estrutura
- TableIdentifier estrutura
- KeySchemaElement estrutura
- PartitionIndex estrutura
- PartitionIndexDescriptor estrutura
- BackfillError estrutura
- IcebergInput estrutura
- OpenTableFormatInput estrutura
- ViewDefinition estrutura
- ViewDefinitionInput estrutura
- ViewRepresentation estrutura
- ViewRepresentationInput estrutura

Estrutura Table

Representa uma coleção de dados relacionados, organizados em colunas e linhas.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome da tabela. Para a compatibilidade com o Hive, ele deve ser inserido todo em letras minúsculas.

- `DatabaseName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no qual os metadados da tabela residem. Para a compatibilidade com o Hive, ele deve ser todo inserido em letras minúsculas.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A descrição da tabela.

- `Owner` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O proprietário da tabela.

- `CreateTime` – Timestamp.

  A hora em que a definição da tabela foi criada no Catálogo de dados.

- `UpdateTime` – Timestamp.

  A última vez que a tabela foi atualizada.

- `LastAccessTime` – Timestamp.

  A última vez que a tabela foi acessada. Ela geralmente é extraída do HDFS e pode não ser confiável.

- `LastAnalyzedTime` – Timestamp.

  A última vez em que as estatísticas da coluna foram computadas para esta tabela.

- `Retention` – Número (inteiro), não mais do que None (Nenhum).

  O tempo de retenção para esta tabela.

- `StorageDescriptor` – Um objeto StorageDescriptor.

  Um descritor de armazenamento contendo informações sobre o armazenamento físico desta tabela.

- `PartitionKeys` – Uma matriz de objetos Coluna.

Uma lista de colunas pela qual a tabela é particionada. Apenas os tipos primitivos são compatíveis como chaves de partição.

Ao criar uma tabela usada pelo Amazon Athena, e você não especificar nenhum `partitionKeys`, você deve definir pelo menos o valor de `partitionKeys` para uma lista vazia. Por exemplo:

```
"PartitionKeys": []
```

- `ViewOriginalText` – String UTF-8 com comprimento não superior a 409.600 bytes.

  Incluído para compatibilidade com o Apache Hive. Não usado no curso normal das AWS Glue operações. Se a tabela for uma`VIRTUAL_VIEW`, determinada Athena configuração codificada em base64.

- `ViewExpandedText` – String UTF-8 com comprimento não superior a 409.600 bytes.

  Incluído para compatibilidade com o Apache Hive. Não usado no curso normal das AWS Glue operações.

- `TableType` – String UTF-8 com comprimento não superior a 255 bytes.

  O tipo dessa tabela. AWS Glue criará tabelas com o `EXTERNAL_TABLE` tipo. Outros serviços, como Athena, podem criar tabelas com tipos de tabela adicionais.

  AWS Glue tipos de tabela relacionados:

  EXTERNAL_TABLE

  Atributo compatível com o Hive: indica uma tabela não gerenciada pelo Hive.

  GOVERNED

  Usado por AWS Lake Formation. O Catálogo AWS Glue de Dados compreende`GOVERNED`.

- `Parameters` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

  Esses pares de chave/valor definem propriedades associadas com a tabela.

- `CreatedBy` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

A pessoa ou entidade que criou a tabela.

- `IsRegisteredWithLakeFormation` – Booleano.

  Indica se a tabela foi registrada com AWS Lake Formation.

- `TargetTable` – Um objeto TableIdentifier.

  Uma estrutura de `TableIdentifier` que descreve uma tabela de destino para vinculação de recursos.

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que a tabela reside.

- `VersionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da versão da tabela.

- `FederatedTable` – Um objeto FederatedTable.

  Uma estrutura `FederatedTable` que referencia uma entidade fora do AWS Glue Data Catalog.

- `ViewDefinition` – Um objeto ViewDefinition.

  Uma estrutura que contém todas as informações que definem a visualização, incluindo o dialeto ou dialetos da visualização e a consulta.

- `IsMultiDialectView` – Booleano.

  Especifica se a visualização oferece suporte aos dialetos SQL de um ou mais mecanismos de consulta diferentes e, portanto, que ela pode ser lida por esses mecanismos.

TableInput estrutura

Uma estrutura usado para definir uma tabela.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome da tabela. Para a compatibilidade com o Hive, as letras são transformadas em minúsculas quando a tabela é armazenada.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A descrição da tabela.

- `Owner` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O proprietário da tabela. Incluído para compatibilidade com o Apache Hive. Não usado no curso normal das AWS Glue operações.

- `LastAccessTime` – Timestamp.

  A última vez que a tabela foi acessada.

- `LastAnalyzedTime` – Timestamp.

  A última vez em que as estatísticas da coluna foram computadas para esta tabela.

- `Retention` – Número (inteiro), não mais do que None (Nenhum).

  O tempo de retenção para esta tabela.

- `StorageDescriptor` – Um objeto StorageDescriptor.

  Um descritor de armazenamento contendo informações sobre o armazenamento físico desta tabela.

- `PartitionKeys` – Uma matriz de objetos Coluna.

  Uma lista de colunas pela qual a tabela é particionada. Apenas os tipos primitivos são compatíveis como chaves de partição.

  Ao criar uma tabela usada pelo Amazon Athena, e você não especificar nenhum `partitionKeys`, você deve definir pelo menos o valor de `partitionKeys` para uma lista vazia. Por exemplo:

  ```
  "PartitionKeys": []
  ```

- `ViewOriginalText` – String UTF-8 com comprimento não superior a 409.600 bytes.

Incluído para compatibilidade com o Apache Hive. Não usado no curso normal das AWS Glue operações. Se a tabela for umaVIRTUAL_VIEW, determinada Athena configuração codificada em base64.

- ViewExpandedText – String UTF-8 com comprimento não superior a 409.600 bytes.

  Incluído para compatibilidade com o Apache Hive. Não usado no curso normal das AWS Glue operações.

- TableType – String UTF-8 com comprimento não superior a 255 bytes.

  O tipo dessa tabela. AWS Glue criará tabelas com o EXTERNAL_TABLE tipo. Outros serviços, como Athena, podem criar tabelas com tipos de tabela adicionais.

  AWS Glue tipos de tabela relacionados:

  EXTERNAL_TABLE

  Atributo compatível com o Hive: indica uma tabela não gerenciada pelo Hive.

  GOVERNED

  Usado por AWS Lake Formation. O Catálogo AWS Glue de Dados compreendeGOVERNED.

- Parameters – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

  Esses pares de chave/valor definem propriedades associadas com a tabela.

- TargetTable – Um objeto TableIdentifier.

  Uma estrutura de TableIdentifier que descreve uma tabela de destino para vinculação de recursos.

- ViewDefinition – Um objeto ViewDefinitionInput.

  Uma estrutura que contém todas as informações que definem a visualização, incluindo o dialeto ou dialetos da visualização e a consulta.

FederatedTable estrutura

Uma tabela que aponta para uma entidade fora do AWS Glue Data Catalog.

Campos

- `Identifier` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Single-line string pattern.

  Um identificador exclusivo da tabela federada.
- `DatabaseIdentifier` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Single-line string pattern.

  Um identificador exclusivo do banco de dados federado.
- `ConnectionName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da conexão com o metastore externo.

Estrutura da coluna

Uma coluna em uma `Table`.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da `Column`.
- `Type` – String UTF-8 com comprimento não superior a 131.072 bytes, correspondente a Single-line string pattern.

  O tipo de dados da `Column`.
- `Comment` – String de comentário, inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um comentário de texto livre.
- `Parameters` – Um array de mapa dos pares de valor-chave.

Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

Esses pares de chave-valor definem propriedades associadas à coluna.

StorageDescriptor estrutura

Descreve o armazenamento físico dos dados da tabela.

Campos

- `Columns` – Uma matriz de objetos Coluna.

  Uma lista de `Columns` na tabela.
- `Location` – String de local, inferior a 2056 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A localização física da tabela. Por padrão, assume o formato da localização do warehouse, seguida pela localização do banco de dados no warehouse, seguida pelo nome da tabela.
- `AdditionalLocations` – Uma matriz de strings UTF-8.

  Uma lista de locais que apontam para o caminho onde uma tabela do Delta está localizada.
- `InputFormat` – String de formato, inferior a 128 bytes de comprimento, correspondente a Single-line string pattern.

  O formato de entrada: `SequenceFileInputFormat` (binário), `TextInputFormat` ou um formato personalizado.
- `OutputFormat` – String de formato, inferior a 128 bytes de comprimento, correspondente a Single-line string pattern.

  O formato de saída: `SequenceFileOutputFormat` (binário), `IgnoreKeyTextOutputFormat` ou um formato personalizado.
- `Compressed` – Booleano.

  `True` se os dados da tabela estiverem compactados, ou `False` se não estiverem.
- `NumberOfBuckets` – Número (íntegro).

Você deverá especificar se a tabela contiver qualquer coluna de dimensão.

- `SerdeInfo` – Um objeto SerDeInfo.

  As informações de serialização/desserialização (). SerDe

- `BucketColumns` – Uma matriz de strings UTF-8.

  Uma lista de colunas de agrupamento, armazenamento em clusters e armazenamento em buckets de reducers na tabela.

- `SortColumns` – Uma matriz de objetos Ordem.

  Uma lista que especifica a ordem de classificação de cada bucket na tabela.

- `Parameters` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

  As propriedades fornecidas pelo usuário em formato de chave/valor.

- `SkewedInfo` – Um objeto SkewedInfo.

  As informações sobre valores que aparecem com frequência em uma coluna (valores distorcidos).

- `StoredAsSubDirectories` – Booleano.

  `True`, se os dados na tabela estiverem armazenados em subdiretórios, ou `False`, caso contrário.

- `SchemaReference` – Um objeto SchemaReference.

  Um objeto que faz referência a um esquema armazenado no Registro do AWS Glue Esquema.

  Ao criar uma tabela, você pode passar uma lista vazia de colunas para o esquema e, em vez disso, usar uma referência de esquema.

SchemaReference estrutura

Um objeto que faz referência a um esquema armazenado no Registro do AWS Glue Esquema.

Campos

- `SchemaId` – Um objeto SchemaId.

  Uma estrutura que contém campos de identidade de esquema. Esse ou o `SchemaVersionId` tem que ser fornecido.

- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O ID exclusivo atribuído a uma versão do esquema. Esse ou o `SchemaId` tem que ser fornecido.

- `SchemaVersionNumber`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  O número da versão do esquema.

SerDeInfo estrutura

Informações sobre um programa de serialização/desserialização (SerDe) que serve como extrator e carregador.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do SerDe.

- `SerializationLibrary` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Normalmente, a classe que implementa o. SerDe Um exemplo é `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe`.

- `Parameters` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

  Esses pares de valores-chave definem parâmetros de inicialização para o. SerDe

Estrutura Order

Especifica a ordem de classificação de uma coluna classificada.

Campos

- `Column` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da coluna.

- `SortOrder` – Obrigatório: Número (inteiro), não mais do que 1.

  Indica se a coluna é classificada em ordem crescente (`== 1`) ou decrescente (`==0`).

SkewedInfo estrutura

Especifica valores distorcidos em uma tabela. Valores distorcidos são aqueles que ocorrem com muita frequência.

Campos

- `SkewedColumnNames` – Uma matriz de strings UTF-8.

  Uma lista de nomes de colunas que contêm valores distorcidos.

- `SkewedColumnValues` – Uma matriz de strings UTF-8.

  Uma lista de valores que aparecem com tanta frequência que devem ser considerados distorcidos.

- `SkewedColumnValueLocationMaps` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Um mapeamento de valores distorcidos para as colunas nas quais estão presentes.

TableVersion estrutura

Especifica a versão de uma tabela.

Campos

- `Table` – Um objeto Tabela.

  A tabela em questão.

- `VersionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O valor de ID que identifica esta versão da tabela. Um `VersionId` é uma representação em string de um inteiro. Cada versão é incrementada em 1.

TableError estrutura

Um registro de erro para operações de tabela.

Campos

- `TableName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela. Para a compatibilidade com o Hive, ele deve ser inserido todo em letras minúsculas.

- `ErrorDetail` – Um objeto ErrorDetail.

  Os detalhes sobre o erro.

TableVersionError estrutura

Um registro de erro para operações da versão da tabela.

Campos

- `TableName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela em questão.

- `VersionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O valor de ID da versão em questão. Um `VersionID` é uma representação em string de um inteiro. Cada versão é incrementada em 1.

- `ErrorDetail` – Um objeto ErrorDetail.

  Os detalhes sobre o erro.

SortCriterion estrutura

Especifica um campo pelo qual classificar e uma ordem de classificação.

Campos

- `FieldName`: valor de string não superior a 1.024 bytes de comprimento.

  O nome do campo no qual classificar.

- `Sort`: string UTF-8 (valores válidos: `ASC="ASCENDING"` | `DESC="DESCENDING"`).

  Uma classificação crescente ou decrescente.

TableIdentifier estrutura

Uma estrutura de que descreve uma tabela de destino para vinculação de recursos.

Campos

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que a tabela reside.

- `DatabaseName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogo que contém a tabela de destino.

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela de destino.

- `Region` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Região da tabela de destino.

KeySchemaElement estrutura

Um par de chaves de partição que consiste em um nome e um tipo.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de uma chave de partição.

- `Type`: obrigatório: string UTF-8 com comprimento não superior a 131.072 bytes, correspondente a Single-line string pattern.

  O tipo de uma chave de partição.

PartitionIndex estrutura

Uma estrutura para um índice de partição.

Campos

- `Keys` – Obrigatório: uma matriz de strings UTF-8, pelo menos 1 string.

  As chaves para o índice de partição.

- `IndexName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do índice de partição.

PartitionIndexDescriptor estrutura

Um descritor para um índice de partição em uma tabela.

Campos

- `IndexName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do índice de partição.

- `Keys`: obrigatório: uma matriz de objetos KeySchemaElement, pelo menos uma estrutura.

  Uma lista de uma ou mais chaves, como estruturas `KeySchemaElement`, para o índice de partição.

- `IndexStatus`: obrigatório: string UTF-8 (valores válidos: `CREATING` | `ACTIVE` | `DELETING` | `FAILED`).

  O status do índice de partição.

  Os possíveis status são:

  - CREATING (CRIANDO): o índice está sendo criado. Quando um índice está em um estado CREATING, o índice ou sua tabela não podem ser excluídos.
  - ACTIVE (ATIVO): a criação do índice foi bem-sucedida.
  - FAILED (FALHOU): a criação do índice falhou.
  - DELETING (EXCLUINDO): o índice é excluído da lista de índices.

- `BackfillErrors` – Uma matriz de objetos BackfillError.

  Uma lista de erros que podem ocorrer no registro de índices de partição para uma tabela existente.

BackfillError estrutura

Uma lista de erros que podem ocorrer no registro de índices de partição para uma tabela existente.

Esses erros fornecem os detalhes sobre por que um registro de índice falhou e fornecem um número limitado de partições na resposta, para que você possa corrigir as partições com falha e tentar registrar o índice novamente. O conjunto mais comum de erros que podem ocorrer são categorizados da seguinte forma:

- EncryptedPartitionError: As partições são criptografadas.
- InvalidPartitionTypeDataError: o valor da partição não corresponde ao tipo de dados dessa coluna de partição.

- MissingPartitionValueError: As partições são criptografadas.
- UnsupportedPartitionCharacterError: caracteres dentro do valor da partição não são suportados. Por exemplo: U+0000, U+0001, U+0002.
- InternalError: Qualquer erro que não pertença a outros códigos de erro.

Campos

- `Code` – String UTF-8 (valores válidos: `ENCRYPTED_PARTITION_ERROR` | `INTERNAL_ERROR` | `INVALID_PARTITION_TYPE_DATA_ERROR` | `MISSING_PARTITION_VALUE_ERROR` | `UNSUPPORTED_PARTITION_CHARACTER_ERROR`).

  O código de erro para um erro que ocorreu durante o registro de índices de partição para uma tabela existente.
- `Partitions` – Uma matriz de objetos PartitionValueList.

  Uma lista com um número limitado de partições na resposta.

IcebergInput estrutura

Uma estrutura que define uma tabela de metadados do Apache Iceberg a ser criada no catálogo.

Campos

- `MetadataOperation`: obrigatório: string UTF-8 (valores válidos: `CREATE`).

  Uma operação de metadados obrigatória. Só pode ser definida como `CREATE`.
- `Version` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  A versão da tabela para a tabela do Iceberg. Padronizado como 2.

OpenTableFormatInput estrutura

Uma estrutura que representa uma tabela de formato aberto.

Campos

- `IcebergInput` – Um objeto IcebergInput.

Especifica uma estrutura `IcebergInput` que define uma tabela de metadados do Apache Iceberg.

ViewDefinition estrutura

Uma estrutura que contém detalhes para representações.

Campos

- `IsProtected` – Booleano.

  É possível definir esse sinalizador como verdadeiro para instruir o mecanismo a não inserir as operações fornecidas pelo usuário no plano lógico da exibição durante o planejamento da consulta. No entanto, definir esse sinalizador não garante que o mecanismo estará em conformidade. Consulte a documentação do mecanismo para entender as garantias fornecidas, se houver.
- `Definer` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Single-line string pattern.

  O definidor de uma visualização em SQL.
- `SubObjects` – Uma matriz de strings UTF-8, no máximo 10 strings.

  Uma lista de nomes do recurso da Amazon (ARNs) da tabela.
- `Representations`: uma matriz de objetos ViewRepresentation, não menos do que 1 ou superior a 1.000 estruturas.

  Uma lista de representações.

ViewDefinitionInput estrutura

Uma estrutura contendo detalhes para criar ou atualizar uma AWS Glue exibição.

Campos

- `IsProtected` – Booleano.

  É possível definir esse sinalizador como verdadeiro para instruir o mecanismo a não inserir as operações fornecidas pelo usuário no plano lógico da exibição durante o planejamento

da consulta. No entanto, definir esse sinalizador não garante que o mecanismo estará em conformidade. Consulte a documentação do mecanismo para entender as garantias fornecidas, se houver.

- `Definer` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Single-line string pattern.

  O definidor de uma visualização em SQL.

- `Representations`: uma matriz de objetos ViewRepresentationInput com não menos que 1 nem mais que 10 estruturas.

  Uma lista de estruturas que contém o dialeto da visualização e a consulta que define a visualização.

- `SubObjects` – Uma matriz de strings UTF-8, no máximo 10 strings.

  Uma lista dos ARNs da tabela base que compõem a visualização.

ViewRepresentation estrutura

Uma estrutura que contém o dialeto da visualização e a consulta que define a visualização.

Campos

- `Dialect` – String UTF-8 (valores válidos: `REDSHIFT` | `ATHENA` | `SPARK`).

  O dialeto do mecanismo de consulta.

- `DialectVersion`: string UTF-8, não menos do que 1 ou superior a 255 bytes de comprimento.

  A versão do dialeto do mecanismo de consulta. Por exemplo, 3.0.0.

- `ViewOriginalText` – String UTF-8 com comprimento não superior a 409.600 bytes.

  A consulta `SELECT` fornecida pelo cliente durante `CREATE VIEW DDL`. Esse SQL não é usado durante uma consulta em uma visualização (`ViewExpandedText` é usado em vez disso). `ViewOriginalText` é usada para casos como `SHOW CREATE VIEW` quando os usuários querem ver o comando DDL original que criou a visualização.

- `ViewExpandedText` – String UTF-8 com comprimento não superior a 409.600 bytes.

  O SQL expandido para a visualização. Esse SQL é usado pelos mecanismos durante o processamento de uma consulta em uma visualização. Os mecanismos podem realizar

operações durante a criação da visualização para transformar `ViewOriginalText` em `ViewExpandedText`. Por exemplo: .

- Identificadores totalmente qualificados: `SELECT * from table1 -> SELECT * from db1.table1`

- `ValidationConnection` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da conexão a ser usada para validar a representação específica da visualização.

- `IsStale` – Booleano.

  Os dialetos marcados como obsoletos não são mais válidos e devem ser atualizados antes que possam ser consultados em seus respectivos mecanismos de consulta.

ViewRepresentationInput estrutura

Uma estrutura que contém detalhes de uma representação para atualizar ou criar uma visualização do Lake Formation.

Campos

- `Dialect` – String UTF-8 (valores válidos: `REDSHIFT` | `ATHENA` | `SPARK`).

  Um parâmetro que especifica o tipo do mecanismo de uma representação específica.

- `DialectVersion`: string UTF-8, não menos do que 1 ou superior a 255 bytes de comprimento.

  Um parâmetro que especifica a versão do mecanismo de uma representação específica.

- `ViewOriginalText` – String UTF-8 com comprimento não superior a 409.600 bytes.

  Uma string que representa a consulta SQL original que descreve a visualização.

- `ValidationConnection` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da conexão a ser usada para validar a representação específica da visualização.

- `ViewExpandedText` – String UTF-8 com comprimento não superior a 409.600 bytes.

  Uma string que representa a consulta SQL que descreve a exibição com ARNs de recursos expandidos

Operações

- CreateTable ação (Python: create_table)
- UpdateTable ação (Python: update_table)
- DeleteTable ação (Python: delete_table)
- BatchDeleteTable ação (Python: batch_delete_table)
- GetTable ação (Python: get_table)
- GetTables ação (Python: get_tables)
- GetTableVersion ação (Python: get_table_version)
- GetTableVersions ação (Python: get_table_versions)
- DeleteTableVersion ação (Python: delete_table_version)
- BatchDeleteTableVersion ação (Python: batch_delete_table_version)
- SearchTables ação (Python: search_tables)
- GetPartitionIndexes ação (Python: get_partition_indexes)
- CreatePartitionIndex ação (Python: create_partition_index)
- DeletePartitionIndex ação (Python: delete_partition_index)
- GetColumnStatisticsForTable ação (Python: get_column_statistics_for_table)
- UpdateColumnStatisticsForTable ação (Python: update_column_statistics_for_table)
- DeleteColumnStatisticsForTable ação (Python: delete_column_statistics_for_table)

CreateTable ação (Python: create_table)

Cria uma definição de nova tabela no catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que a `Table` será criada. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O banco de dados do catálogo em que a nova tabela será criada. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.

- `TableInput` – Obrigatório: um objeto TableInput.

  O objeto `TableInput` que define a tabela de metadados a ser criada no catálogo.

- `PartitionIndexes`: uma matriz de objetos PartitionIndex, não mais do que três estruturas.

  Uma lista de índices de partição, estruturas `PartitionIndex`, para criar na tabela.

- `TransactionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #16.

  O ID da transação.

- `OpenTableFormatInput` – Um objeto OpenTableFormatInput.

  Especifica uma estrutura `OpenTableFormatInput` ao criar uma tabela de formato aberto.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `AlreadyExistsException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`
- `ResourceNotReadyException`

UpdateTable ação (Python: update_table)

Atualiza uma tabela de metadados no catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual reside a tabela. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados do catálogo no qual a tabela reside. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.
- `TableInput` – Obrigatório: um objeto TableInput.

  O objeto `TableInput` atualizado que define a tabela de metadados no catálogo.
- `SkipArchive` – Booleano.

  Por padrão, o `UpdateTable` sempre cria uma versão arquivada da tabela antes de atualizá-la. No entanto, se `skipArchive` estiver definido como true, `UpdateTable` não cria a versão arquivada.
- `TransactionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #16.

  O ID da transação na qual atualizar o conteúdo da tabela.
- `VersionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da versão na qual atualizar o conteúdo da tabela.
- `ViewUpdateAction` – String UTF-8 (valores válidos: `ADD` | `REPLACE` | `ADD_OR_REPLACE` | `DROP`).

  A operação a ser executada ao atualizar a visualização.
- `Force` – Booleano.

  Um sinalizador que pode ser definido como verdadeiro para ignorar os requisitos correspondentes de correspondência do descritor de armazenamento e do subobjeto.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`
- `ResourceNotReadyException`

DeleteTable ação (Python: delete_table)

Remove uma definição de tabela do catálogo de dados.

> **Note**
>
> Depois de concluir essa operação, você não terá mais acesso às partições e versões de tabela que pertençam à tabela excluída. O AWS Glue exclui esses recursos "órfãos" de forma assíncrona e pontual, a critério do serviço.
> Para garantir a exclusão imediata de todos os recursos relacionados, antes de chamar `DeleteTable`, use `DeleteTableVersion` ou `BatchDeleteTableVersion`, e `DeletePartition` ou `BatchDeletePartition`, para excluir qualquer recurso que pertence à tabela.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual reside a tabela. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome do banco de dados do catálogo no qual a tabela reside. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome da tabela a ser excluída. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.

- `TransactionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #16.

  O ID da transação na qual excluir o conteúdo da tabela.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`
- `ResourceNotReadyException`

BatchDeleteTable ação (Python: batch_delete_table)

Exclui várias tabelas ao mesmo tempo.

> **Note**
>
> Depois de concluir essa operação, você não terá mais acesso às partições e versões de tabela que pertençam à tabela excluída. O AWS Glue exclui esses recursos "órfãos" de forma assíncrona e pontual, a critério do serviço.

Para garantir a exclusão imediata de todos os recursos relacionados, antes de chamar `BatchDeleteTable`, use `DeleteTableVersion` ou `BatchDeleteTableVersion`, e `DeletePartition` ou `BatchDeletePartition`, para excluir qualquer recurso que pertence à tabela.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual reside a tabela. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogo no qual as tabelas a serem excluídas residem. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.

- `TablesToDelete` – Obrigatório: uma matriz de strings UTF-8, no máximo 100 strings.

  Uma lista da tabela a ser excluída.

- `TransactionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #16.

  O ID da transação na qual excluir o conteúdo da tabela.

Resposta

- `Errors` – Uma matriz de objetos TableError.

  Uma lista de erros encontrados na tentativa de excluir as tabelas especificadas.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`

- `OperationTimeoutException`
- `GlueEncryptionException`
- `ResourceNotReadyException`

GetTable ação (Python: get_table)

Recupera a definição da `Table` em um catálogo de dados para uma tabela especificada.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual reside a tabela. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no catálogo em que a tabela reside. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.
- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela para o qual recuperar a definição. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.
- `TransactionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #16.

  O ID da transação na qual ler o conteúdo da tabela.
- `QueryAsOfTime` – Timestamp.

  A hora de ler o conteúdo da tabela. Se não estiver definido, o tempo de confirmação de transação mais recente será usado. Não pode ser especificado junto com `TransactionId`.

Resposta

- `Table` – Um objeto Tabela.

O objeto `Table` que define a tabela especificada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ResourceNotReadyException`
- `FederationSourceException`
- `FederationSourceRetryableException`

GetTables ação (Python: get_tables)

Recupera as definições de algumas ou todas as tabelas em um determinado `Database`.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as tabelas. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O banco de dados no catálogo cujas tabelas serão listadas. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.
- `Expression` – String UTF-8 com comprimento não superior a 2048 bytes, correspondente a Single-line string pattern.

  Um padrão de expressão regular. Se presente, apenas as tabelas cujos nomes correspondem ao padrão serão retornadas.
- `NextToken` – String UTF-8.

Um token de continuação, incluído se esta for uma chamada de continuação.

- `MaxResults`: número (inteiro), não menos do que 1 ou superior a 100.

  O número máximo de tabelas a serem retornados em uma única resposta.

- `TransactionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #16.

  O ID da transação na qual ler o conteúdo da tabela.

- `QueryAsOfTime` – Timestamp.

  A hora de ler o conteúdo da tabela. Se não estiver definido, o tempo de confirmação de transação mais recente será usado. Não pode ser especificado junto com `TransactionId`.

Resposta

- `TableList` – Uma matriz de objetos Tabela.

  Uma lista de objetos `Table` solicitados.

- `NextToken` – String UTF-8.

  Um token de continuação, presente se o segmento de lista atual não for o último.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `FederationSourceException`
- `FederationSourceRetryableException`

GetTableVersion ação (Python: get_table_version)

Recupera uma versão especificada de uma tabela.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as tabelas. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O banco de dados no catálogo em que a tabela reside. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.

- `VersionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O valor de ID da versão de tabela a ser recuperada. Um `VersionID` é uma representação em string de um inteiro. Cada versão é incrementada em 1.

Resposta

- `TableVersion` – Um objeto TableVersion.

  A versão da tabela solicitada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

GetTableVersions ação (Python: get_table_versions)

Recupera uma lista de strings que identificam as versões disponíveis de uma tabela especificada.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as tabelas. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O banco de dados no catálogo em que a tabela reside. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.

- `NextToken` – String UTF-8.

  Um token de continuação, se esta não for a primeira chamada.

- `MaxResults`: número (inteiro), não menos do que 1 ou superior a 100.

  O número máximo de versões de tabela a ser retornado em uma resposta.

Resposta

- `TableVersions` – Uma matriz de objetos TableVersion.

  Uma lista de strings identificando versões disponíveis da tabela especificada.

- `NextToken` – String UTF-8.

  Um token de continuação, se a lista de versões disponíveis não incluir o último token.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeleteTableVersion ação (Python: delete_table_version)

Exclui uma versão especificada de uma tabela.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as tabelas. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O banco de dados no catálogo em que a tabela reside. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.
- `VersionId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da versão de tabela a ser excluída. Um `VersionID` é uma representação em string de um inteiro. Cada versão é incrementada em 1.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

BatchDeleteTableVersion ação (Python: batch_delete_table_version)

Exclui um lote especificado de versões de uma tabela.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as tabelas. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O banco de dados no catálogo em que a tabela reside. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela. Para a compatibilidade com o Hive, o nome deve ser inserido todo em letras minúsculas.
- `VersionIds` – Obrigatório: uma matriz de strings UTF-8, no máximo 100 strings.

  Uma lista dos IDs das versões a serem excluídas. Um `VersionId` é uma representação em string de um inteiro. Cada versão é incrementada em 1.

Resposta

- `Errors` – Uma matriz de objetos TableVersionError.

  Uma lista de erros encontrados durante a tentativa de excluir as versões de tabela especificadas.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

SearchTables ação (Python: search_tables)

Pesquisa um conjunto de tabelas com base nas propriedades dos metadados da tabela, bem como no banco de dados pai. Você pode pesquisar condições de texto ou filtro.

Você só pode obter tabelas às quais tem acesso com base nas políticas de segurança definidas no Lake Formation. Você precisa de pelo menos um acesso somente leitura à tabela para que ela seja retornada. Se você não tiver acesso a todas as colunas na tabela, essas colunas não serão pesquisadas ao retornar a lista de tabelas de volta para você. Se você tiver acesso às colunas, mas não aos dados nas colunas, essas colunas e os metadados associados a essas colunas serão incluídos na pesquisa.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um identificador exclusivo, que consiste em *`account_id`*.
- `NextToken` – String UTF-8.

  Um token de continuação, incluído se esta for uma chamada de continuação.
- `Filters` – Uma matriz de objetos PropertyPredicate.

  Uma lista de pares de chave/valor e um comparador usado para filtrar os resultados da pesquisa. Retorna todas as entidades correspondentes ao predicado.

O membro `Comparator` da estrutura `PropertyPredicate` é usado apenas para campos de tempo, e pode ser omitido para outros tipos de campo. Além disso, ao comparar valores de string, como quando `Key=Name`, um algoritmo de correspondência difusa é usado. O campo `Key` (por exemplo, o valor do campo `Name`) é dividido em tokens em certos caracteres de pontuação, por exemplo, -, :, # etc. Em seguida, cada token é comparado por correspondência exata com o membro `Value` de `PropertyPredicate`. Por exemplo, se `Key=Name` e `Value=link`, as tabelas denominadas `customer-link` e `xx-link-yy` são retornadas, mas `xxlinkyy` não é.

- `SearchText`: valor de string não superior a 1.024 bytes de comprimento.

  Uma string usada para uma pesquisa de texto.

  Especificar um valor em filtros de aspas com base em uma correspondência exata com o valor.

- `SortCriteria`: uma matriz de objetos SortCriterion, não mais do que uma estrutura.

  Uma lista de critérios para classificar os resultados por um nome de campo, em uma ordem crescente ou decrescente.

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de tabelas a serem retornados em uma única resposta.

- `ResourceShareType` – String UTF-8 (valores válidos: `FOREIGN` | `ALL` | `FEDERATED`).

  Permite que você especifique que deseja pesquisar as tabelas compartilhadas com sua conta. Os valores permitidos são `FOREIGN` ou `ALL`.

  - Se definido como `FOREIGN`, listará as tabelas compartilhadas com sua conta.
  - Se definido como `ALL`, listará as tabelas compartilhadas com sua conta, bem como as tabelas em sua conta local.

Resposta

- `NextToken` – String UTF-8.

  Um token de continuação, presente se o segmento de lista atual não for o último.

- `TableList` – Uma matriz de objetos Tabela.

  Uma lista de objetos `Table` solicitados. A resposta `SearchTables` retorna apenas as tabelas às quais você tem acesso.

Erros

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

GetPartitionIndexes ação (Python: get_partition_indexes)

Recupera os índices de partição associados a uma tabela.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo no qual reside a tabela.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Especifica o nome de um banco de dados a partir do qual você deseja recuperar índices de partição.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Especifica o nome de uma tabela para a qual você deseja recuperar os índices da partição.
- `NextToken` – String UTF-8.

  Um token de continuação, incluído se esta for uma chamada de continuação.

Resposta

- `PartitionIndexDescriptorList` – Uma matriz de objetos PartitionIndexDescriptor.

  Uma lista de descritores de índice.
- `NextToken` – String UTF-8.

  Um token de continuação, presente se o segmento de lista atual não for o último.

Erros

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ConflictException`

CreatePartitionIndex ação (Python: create_partition_index)

Cria um índice de partição especificado em uma tabela existente.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo no qual reside a tabela.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Especifica o nome de um banco de dados no qual você deseja criar um índice de partição.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Especifica o nome de uma tabela na qual você deseja criar um índice de partição.
- `PartitionIndex` – Obrigatório: um objeto PartitionIndex.

  Especifica uma estrutura `PartitionIndex` para criar um índice de partição em uma tabela existente.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `AlreadyExistsException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeletePartitionIndex ação (Python: delete_partition_index)

Exclui um índice de partição especificado de uma tabela existente.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo no qual reside a tabela.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Especifica o nome de um banco de dados do qual você deseja excluir um índice de partição.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Especifica o nome de uma tabela da qual você deseja excluir um índice de partição.
- `IndexName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do índice de partição a ser excluído.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ConflictException`
- `GlueEncryptionException`

GetColumnStatisticsForTable ação (Python: get_column_statistics_for_table)

Recupera estatísticas de tabelas das colunas.

A permissão de Identity and Access Management (IAM) necessária para essa operação é `GetTable`.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as partições em questão. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogos no qual as partições residem.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela das partições.
- `ColumnNames` – Obrigatório: uma matriz de strings UTF-8, no máximo 100 strings.

  Uma lista de nomes de coluna.

Resposta

- `ColumnStatisticsList` – Uma matriz de objetos ColumnStatistics.

Lista de ColumnStatistics.

- `Errors` – Uma matriz de objetos ColumnError.

  A lista ColumnStatistics disso não foi recuperada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

UpdateColumnStatisticsForTable ação (Python: update_column_statistics_for_table)

Cria ou atualiza estatísticas de tabela das colunas.

A permissão de Identity and Access Management (IAM) necessária para essa operação é `UpdateTable`.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as partições em questão. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogos no qual as partições residem.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela das partições.
- `ColumnStatisticsList` – Obrigatório: uma matriz de objetos ColumnStatistics, não mais de 25 estruturas.

Uma lista de estatísticas de coluna.

Resposta

- `Errors` – Uma matriz de objetos ColumnStatisticsError.

  Lista de ColumnStatisticsErrors.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeleteColumnStatisticsForTable ação (Python: delete_column_statistics_for_table)

Recupera estatísticas de tabelas das colunas.

A permissão de Identity and Access Management (IAM) necessária para essa operação é `DeleteTable`.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as partições em questão. Se nenhum for fornecido, o ID da AWS conta será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogos no qual as partições residem.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome da tabela das partições.

- `ColumnName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da coluna.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

API de partição

A Partição da API descreve os tipos de dados e operações usadas para trabalhar com partições.

Tipos de dados

- Estrutura Partition
- Estrutura PartitionInput
- Estrutura PartitionSpecWithSharedStorageDescriptor
- Estrutura PartitionListComposingSpec
- Estrutura PartitionSpecProxy
- Estrutura PartitionValueList
- Estrutura Segment
- Estrutura PartitionError
- Estrutura BatchUpdatePartitionFailureEntry

- Estrutura BatchUpdatePartitionRequestEntry
- Estrutura StorageDescriptor
- Estrutura SchemaReference
- Estrutura SerDeInfo
- Estrutura SkewedInfo

Estrutura Partition

Representa uma fatia dos dados da tabela.

Campos

- `Values` – Uma matriz de strings UTF-8.

  Os valores da partição.
- `DatabaseName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogo no qual a função será criada.
- `TableName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela do banco de dados na qual criar a partição.
- `CreationTime` – Timestamp.

  A hora em que a partição foi criada.
- `LastAccessTime` – Timestamp.

  A hora em que a partição foi acessada pela última vez.
- `StorageDescriptor` – Um objeto StorageDescriptor.

  Fornece informações sobre o local físico no qual a partição está armazenada.
- `Parameters` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

Esses pares de chave/valor definem parâmetros de partição.

- `LastAnalyzedTime` – Timestamp.

  A última vez em que as estatísticas da coluna foram computadas para esta partição.

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do Data Catalog em que a partição reside.

Estrutura PartitionInput

A estrutura usada para criar e atualizar uma partição.

Campos

- `Values` – Uma matriz de strings UTF-8.

  Os valores da partição. Embora este parâmetro não seja exigido pelo SDK, você deve especificá-lo para uma entrada válida.

  Os valores das chaves da nova partição devem ser passados como uma matriz de objetos String que devem ser classificados na mesma ordem que as chaves de partição que aparecem no prefixo do Amazon S3. Caso contrário, o AWS Glue adicionará os valores às chaves incorretas.

- `LastAccessTime` – Timestamp.

  A hora em que a partição foi acessada pela última vez.

- `StorageDescriptor` – Um objeto StorageDescriptor.

  Fornece informações sobre o local físico no qual a partição está armazenada.

- `Parameters` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

  Esses pares de chave/valor definem parâmetros de partição.

- `LastAnalyzedTime` – Timestamp.

A última vez em que as estatísticas da coluna foram computadas para esta partição.

Estrutura PartitionSpecWithSharedStorageDescriptor

Uma especificação de partição para partições que compartilham um local físico.

Campos

- `StorageDescriptor` – Um objeto StorageDescriptor.

  As informações do armazenamento físico compartilhado.
- `Partitions` – Uma matriz de objetos Partition.

  Uma lista das partições que compartilham este local físico.

Estrutura PartitionListComposingSpec

Lista as partições relacionadas.

Campos

- `Partitions` – Uma matriz de objetos Partition.

  Uma lista das partições na especificação de composição.

Estrutura PartitionSpecProxy

Fornece um caminho raiz para as partições especificadas.

Campos

- `DatabaseName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O banco de dados de catálogo no qual residem as partições.
- `TableName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela que contém as partições.

- `RootPath` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O caminho raiz do proxy para as partições.
- `PartitionSpecWithSharedSD` – Um objeto PartitionSpecWithSharedStorageDescriptor.

  Uma especificação de partições que compartilham o mesmo local de armazenamento físico.
- `PartitionListComposingSpec` – Um objeto PartitionListComposingSpec.

  Especifica uma lista de partições.

Estrutura PartitionValueList

Contém uma lista dos valores que definem as partições.

Campos

- `Values`: obrigatório: uma matriz de strings UTF-8.

  A lista de valores.

Estrutura Segment

Define uma região não sobreposta das partições de uma tabela, permitindo que várias solicitações sejam executadas simultaneamente.

Campos

- `SegmentNumber` – Obrigatório: número (inteiro), não mais do que Nenhum.

  O número de índice baseado em zero do segmento. Por exemplo, se o número total de segmentos for 4, os valores de `SegmentNumber` vão variar de 0 a 3.
- `TotalSegments` – Obrigatório: número (inteiro), no mínimo 1 ou mais que 10.

  O número total de segmentos.

Estrutura PartitionError

Contém informações sobre um erro de partição.

Campos

- `PartitionValues` – Uma matriz de strings UTF-8.

  Os valores que definem a partição.
- `ErrorDetail` – Um objeto ErrorDetail.

  Os detalhes sobre o erro de partição.

Estrutura BatchUpdatePartitionFailureEntry

Contém informações sobre um erro de partição de atualização em lote.

Campos

- `PartitionValueList` – Uma matriz de strings UTF-8, no máximo 100 strings.

  Uma lista dos valores que definem as partições.
- `ErrorDetail` – Um objeto ErrorDetail.

  Os detalhes sobre o erro de partição de atualização em lote.

Estrutura BatchUpdatePartitionRequestEntry

Uma estrutura que contém os valores e a estrutura usados para atualizar uma partição.

Campos

- `PartitionValueList` – Obrigatório: uma matriz de strings UTF-8, no máximo 100 strings.

  Uma lista dos valores que definem as partições.
- `PartitionInput` – Obrigatório: um objeto PartitionInput.

  A estrutura usada para atualizar uma partição.

Estrutura StorageDescriptor

Descreve o armazenamento físico dos dados da tabela.

Campos

- `Columns` – Uma matriz de objetos Coluna.

  Uma lista de `Columns` na tabela.

- `Location` – String de local, inferior a 2056 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A localização física da tabela. Por padrão, assume o formato da localização do warehouse, seguida pela localização do banco de dados no warehouse, seguida pelo nome da tabela.

- `AdditionalLocations` – Uma matriz de strings UTF-8.

  Uma lista de locais que apontam para o caminho onde uma tabela do Delta está localizada.

- `InputFormat` – String de formato, inferior a 128 bytes de comprimento, correspondente a Single-line string pattern.

  O formato de entrada: `SequenceFileInputFormat` (binário), `TextInputFormat` ou um formato personalizado.

- `OutputFormat` – String de formato, inferior a 128 bytes de comprimento, correspondente a Single-line string pattern.

  O formato de saída: `SequenceFileOutputFormat` (binário), `IgnoreKeyTextOutputFormat` ou um formato personalizado.

- `Compressed` – Booleano.

  `True` se os dados da tabela estiverem compactados, ou `False` se não estiverem.

- `NumberOfBuckets` – Número (íntegro).

  Você deverá especificar se a tabela contiver qualquer coluna de dimensão.

- `SerdeInfo` – Um objeto SerDeInfo.

  As informações de serialização/desserialização (SerDe).

- `BucketColumns` – Uma matriz de strings UTF-8.

  Uma lista de colunas de agrupamento, armazenamento em clusters e armazenamento em buckets de reducers na tabela.

- `SortColumns` – Uma matriz de objetos Ordem.

Uma lista que especifica a ordem de classificação de cada bucket na tabela.

- `Parameters` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

  As propriedades fornecidas pelo usuário em formato de chave/valor.
- `SkewedInfo` – Um objeto SkewedInfo.

  As informações sobre valores que aparecem com frequência em uma coluna (valores distorcidos).
- `StoredAsSubDirectories` – Booleano.

  `True`, se os dados na tabela estiverem armazenados em subdiretórios, ou `False`, caso contrário.
- `SchemaReference` – Um objeto SchemaReference.

  Um objeto que faz referência a um esquema armazenado no AWS Glue Schema Registry.

  Ao criar uma tabela, você pode passar uma lista vazia de colunas para o esquema e, em vez disso, usar uma referência de esquema.

Estrutura SchemaReference

Um objeto que faz referência a um esquema armazenado no AWS Glue Schema Registry.

Campos

- `SchemaId` – Um objeto SchemaId.

  Uma estrutura que contém campos de identidade de esquema. Esse ou o `SchemaVersionId` tem que ser fornecido.
- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O ID exclusivo atribuído a uma versão do esquema. Esse ou o `SchemaId` tem que ser fornecido.
- `SchemaVersionNumber`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  O número da versão do esquema.

Estrutura SerDeInfo

As informações sobre um programa de serialização/desserialização (SerDe), que serve como extrator e carregador.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome de SerDe.

- `SerializationLibrary` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Normalmente, é a classe que implementa o SerDe. Um exemplo é `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe`.

- `Parameters` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string de chave, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Cada valor é uma string UTF-8 com comprimento não superior a 512.000 bytes.

  Esses pares de chave/valor definem parâmetros de inicialização para SerDe.

Estrutura SkewedInfo

Especifica valores distorcidos em uma tabela. Valores distorcidos são aqueles que ocorrem com muita frequência.

Campos

- `SkewedColumnNames` – Uma matriz de strings UTF-8.

  Uma lista de nomes de colunas que contêm valores distorcidos.

- `SkewedColumnValues` – Uma matriz de strings UTF-8.

  Uma lista de valores que aparecem com tanta frequência que devem ser considerados distorcidos.

- `SkewedColumnValueLocationMaps` – Um array de mapa dos pares de valor-chave.

Cada chave é uma sequência de caracteres UTF-8.

Cada valor é uma sequência de caracteres UTF-8.

Um mapeamento de valores distorcidos para as colunas nas quais estão presentes.

Operações

- Ação CreatePartition (Python: create_partition)
- Ação BatchCreatePartition (Python: batch_create_partition)
- Ação UpdatePartition (Python: update_partition)
- Ação DeletePartition (Python: delete_partition)
- Ação BatchDeletePartition (Python: batch_delete_partition)
- Ação GetPartition (Python: get_partition)
- Ação GetPartitions (Python: get_partitions)
- Ação BatchGetPartition (Python: batch_get_partition)
- Ação BatchUpdatePartition (Python: batch_update_partition)
- Ação GetColumnStatisticsForPartition (Python: get_column_statistics_for_partition)
- Ação UpdateColumnStatisticsForPartition (Python: update_column_statistics_for_partition)
- Ação DeleteColumnStatisticsForPartition (Python: delete_column_statistics_for_partition)

Ação CreatePartition (Python: create_partition)

Cria uma nova partição.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da conta da AWS do catálogo no qual a partição deve ser criada.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de metadados no qual a partição deve ser criada.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela de metadados na qual a partição deve ser criada.

- `PartitionInput` – Obrigatório: um objeto PartitionInput.

  Uma estrutura `PartitionInput` que define a partição a ser criada.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Ação BatchCreatePartition (Python: batch_create_partition)

Cria uma ou mais partições em uma operação em lote.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo no qual a partição deve ser criada. No momento, deve ser o ID da conta da AWS.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de metadados no qual a partição deve ser criada.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela de metadados na qual a partição deve ser criada.

- `PartitionInputList` – Obrigatório: uma matriz de objetos PartitionInput, não mais de 100 estruturas.

  Uma lista das estruturas `PartitionInput` que definem as partições a serem criadas.

Resposta

- `Errors` – Uma matriz de objetos PartitionError.

  Os erros encontrados ao tentar criar as partições solicitadas.

Erros

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Ação UpdatePartition (Python: update_partition)

Atualiza uma partição.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual reside a partição a ser atualizada. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados do catálogo no qual a tabela em questão reside.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela de metadados na qual a partição a ser atualizada está localizada.

- `PartitionValueList` – Obrigatório: uma matriz de strings UTF-8, no máximo 100 strings.

  Lista de valores de chaves-valor de partição que definem a partição a ser atualizada.

- `PartitionInput` – Obrigatório: um objeto PartitionInput.

  O novo objeto de partição para o qual a partição será atualizada.

  A propriedade `Values` não pode ser alterada. Se você quiser alterar as chaves-valor de uma partição, exclua e recrie a partição.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Ação DeletePartition (Python: delete_partition)

Exclui uma partição especificada.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual reside a partição a ser excluída. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados do catálogo no qual a tabela em questão reside.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela que contém a partição a ser excluída.
- `PartitionValues`: obrigatório: uma matriz de strings UTF-8.

  Os valores que definem a partição.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação BatchDeletePartition (Python: batch_delete_partition)

Exclui uma ou mais partições em uma operação em lote.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O ID do catálogo de dados no qual reside a partição a ser excluída. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados do catálogo no qual a tabela em questão reside.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela que contém as partições a serem excluídas.

- `PartitionsToDelete` – Obrigatório: uma matriz de objetos PartitionValueList, não mais de 25 estruturas.

  Uma lista das estruturas `PartitionInput` que definem as partições a serem excluídas.

Resposta

- `Errors` – Uma matriz de objetos PartitionError.

  Os erros encontrados ao tentar excluir as partições solicitadas.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetPartition (Python: get_partition)

Recupera informações sobre uma partição especificada.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O ID do catálogo de dados no qual reside a partição em questão. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogos no qual a partição reside.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela da partição.

- `PartitionValues`: obrigatório: uma matriz de strings UTF-8.

  Os valores que definem a partição.

Resposta

- `Partition` – Um objeto Partition.

  A informação solicitada, na forma de um objeto `Partition`.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `FederationSourceException`
- `FederationSourceRetryableException`

Ação GetPartitions (Python: get_partitions)

Recupera informações sobre as partições em uma tabela.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as partições em questão. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogos no qual as partições residem.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela das partições.

- `Expression` – String de predicado, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma expressão que filtra as partições a serem retornadas.

  A expressão usa sintaxe SQL semelhante à cláusula de filtro SQL `WHERE`. O analisador de instrução SQL JSQLParser analisa a expressão.

  Operadores: estes são os operadores que você pode usar na chamada de API `Expression`:

  =

  Verifica se os valores dos dois operandos são iguais. Em caso afirmativo, a condição é verdadeira.

  Exemplo: suponha que a "variável a" seja 10 e a "variável b" 20.

  (a = b) não é verdadeiro.

  < >

  Verifica se os valores dos dois operandos são iguais. Se os valores não forem iguais, a condição será verdadeira.

  Exemplo: (a < > b) é verdadeiro.

>

Verifica se o valor do operando esquerdo é maior que o valor do operando direito. Em caso afirmativo, a condição é verdadeira.

Exemplo: (a **>** b) não é verdadeiro.

<

Verifica se o valor do operando esquerdo é menor que o valor do operando direito. Em caso afirmativo, a condição é verdadeira.

Exemplo: (a **<** b) é verdadeiro.

>=

Verifica se o valor do operando esquerdo é maior ou igual ao valor do operando direito. Em caso afirmativo, a condição é verdadeira.

Exemplo: (a **>=** b) não é verdadeiro.

<=

Verifica se o valor do operando esquerdo é menor ou igual ao valor do operando direito. Em caso afirmativo, a condição é verdadeira.

Exemplo: (a {**<=** b) é verdadeiro.

AND, OR, IN, BETWEEN, LIKE, NOT, IS NULL

Operadores lógicos.

Tipos de chave de partição compatíveis: veja a seguir as chaves de partição compatíveis.

- `string`
- `date`
- `timestamp`
- `int`
- `bigint`
- `long`
- `tinyint`
- `smallint`

- `decimal`

Se um tipo que não seja válido for encontrado, uma exceção será lançada.

A lista a seguir mostra os operadores válidos em cada tipo. Quando você define um crawler, o tipo `partitionKey` é criado como um `STRING`, para ser compatível com as partições do catálogo.

Exemplo de chamada de API:

Example

A tabela `twitter_partition` tem três partições:

```
year = 2015
        year = 2016
        year = 2017
```

Example

Obter `year` da partição igual a 2015

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year*=*'2015'"
```

Example

Obter `year` da partição entre 2016 e 2018 (exclusivo)

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year>'2016' AND year<'2018'"
```

Example

Obtenha o `year` da partição entre 2015 e 2018 (inclusivo). As seguintes chamadas de API são equivalentes entre si:

```
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

Um filtro de partição curinga, onde a saída de chamada a seguir terá o ano de partição = 2017. Uma expressão regular não é suportada em `LIKE`.

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year LIKE '%7'"
```

- `NextToken` – String UTF-8.

  Um token de continuação, se esta não for a primeira chamada para recuperar essas partições.

- `Segment` – Um objeto Segment.

  O segmento de partições da tabela a ser verificado nesta solicitação.

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de partições a serem retornados em uma única resposta.

- `ExcludeColumnSchema` – Booleano.

  Quando verdadeiro, especifica não retornar o esquema da coluna de partição. Útil quando você está interessado apenas em outros atributos de partição, como valores ou localização de partição. Essa abordagem evita o problema de uma resposta grande ao não retornar dados duplicados.

- `TransactionId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #16.

  O ID da transação na qual ler o conteúdo da partição.

- `QueryAsOfTime` – Timestamp.

  A hora de ler o conteúdo da partição. Se não estiver definido, o tempo de confirmação de transação mais recente será usado. Não pode ser especificado junto com `TransactionId`.

Resposta

- `Partitions` – Uma matriz de objetos Partition.

  Uma lista de partições solicitadas.

- `NextToken` – String UTF-8.

  Um token de continuação, se a lista de partições retornada não incluir o último token.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `InvalidStateException`
- `ResourceNotReadyException`
- `FederationSourceException`
- `FederationSourceRetryableException`

Ação BatchGetPartition (Python: batch_get_partition)

Recupera partições em uma solicitação de lote.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as partições em questão. Se nenhum valor for fornecido, o ID da conta da AWS será usado por padrão.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogos no qual as partições residem.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela das partições.
- `PartitionsToGet` – Obrigatório: uma matriz de objetos PartitionValueList, não mais de 1000 estruturas.

  Uma lista de valores de partição que identifica as partições a serem recuperadas.

Resposta

- `Partitions` – Uma matriz de objetos Partition.

  Uma lista das partições solicitadas.
- `UnprocessedKeys` – Uma matriz de objetos PartitionValueList, não mais de 1000 estruturas.

  Uma lista dos valores de partição na solicitação para a qual as partições não foram retornadas.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `InvalidStateException`
- `FederationSourceException`
- `FederationSourceRetryableException`

Ação BatchUpdatePartition (Python: batch_update_partition)

Atualiza uma ou mais partições em uma operação em lote.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O ID de catálogo no qual a partição deve ser atualizada. No momento, deve ser o ID da conta da AWS.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de metadados no qual a partição deve ser atualizada.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela de metadados na qual a partição deve ser atualizada.

- `Entries`: obrigatório: uma matriz de objetos BatchUpdatePartitionRequestEntry, com número de estruturas entre 1 e 100.

  Uma lista de até 100 objetos `BatchUpdatePartitionRequestEntry` a serem atualizados.

Resposta

- `Errors` – Uma matriz de objetos BatchUpdatePartitionFailureEntry.

  Os erros encontrados ao tentar atualizar as partições solicitadas. Uma lista dos objetos `BatchUpdatePartitionFailureEntry`.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`

Ação GetColumnStatisticsForPartition (Python: get_column_statistics_for_partition)

Recupera estatísticas de partição das colunas.

A permissão de Identity and Access Management (IAM) necessária para essa operação é `GetPartition`.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as partições em questão. Se nenhum valor for fornecido, o ID da conta da AWS será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogos no qual as partições residem.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela das partições.
- `PartitionValues`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de valores de partição que identifica a partição.
- `ColumnNames` – Obrigatório: uma matriz de strings UTF-8, no máximo 100 strings.

  Uma lista de nomes de coluna.

Resposta

- `ColumnStatisticsList` – Uma matriz de objetos ColumnStatistics.

  Lista de ColumnStatistics cuja recuperação falhou.
- `Errors` – Uma matriz de objetos ColumnError.

  Ocorreu um erro durante a recuperação de dados de estatísticas de coluna.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

- `GlueEncryptionException`

Ação UpdateColumnStatisticsForPartition (Python: update_column_statistics_for_partition)

Cria ou atualiza estatísticas de partição de colunas.

A permissão de Identity and Access Management (IAM) necessária para essa operação é `UpdatePartition`.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as partições em questão. Se nenhum valor for fornecido, o ID da conta da AWS será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogos no qual as partições residem.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela das partições.
- `PartitionValues`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de valores de partição que identifica a partição.
- `ColumnStatisticsList` – Obrigatório: uma matriz de objetos ColumnStatistics, não mais de 25 estruturas.

  Uma lista de estatísticas de coluna.

Resposta

- `Errors` – Uma matriz de objetos ColumnStatisticsError.

  Ocorreu um erro durante a atualização de dados de estatísticas de coluna

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Ação DeleteColumnStatisticsForPartition (Python: delete_column_statistics_for_partition)

Exclui as estatísticas de coluna de partição de uma coluna.

A permissão de Identity and Access Management (IAM) necessária para essa operação é `DeletePartition`.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual residem as partições em questão. Se nenhum valor for fornecido, o ID da conta da AWS será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogos no qual as partições residem.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela das partições.
- `PartitionValues`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de valores de partição que identifica a partição.
- `ColumnName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da coluna.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

API de conexão

A API de conexão descreve os tipos de dados de conexão do AWS Glue e a API para criar, excluir, atualizar e listar conexões.

Tipos de dados

- Estrutura Connection
- Estrutura ConnectionInput
- Estrutura PhysicalConnectionRequirements
- Estrutura GetConnectionsFilter

Estrutura Connection

Define uma conexão para uma fonte de dados.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição da conexão.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

A descrição da conexão.

- `ConnectionType` – String UTF-8 (valores válidos: `JDBC` | `SFTP` | `MONGODB` | `KAFKA` | `NETWORK` | `MARKETPLACE` | `CUSTOM` | `SALESFORCE`).

  O tipo da conexão. No momento, o SFTP não é compatível.

- `MatchCriteria` – Uma matriz de strings UTF-8, no máximo 10 strings.

  Uma lista de critérios que podem ser usados na seleção dessa conexão.

- `ConnectionProperties`: uma matriz de mapa dos pares de chave-valor, não mais do que 100 pares.

  Cada chave é uma string 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 é um string de valor, com não superior a 1024 bytes.

  Esses pares de chave-valor definem parâmetros para a conexão:

  - `HOST` - O URI do host: o nome de domínio totalmente qualificado (FQDN) ou o endereço IPv4 do host do banco de dados.
  - `PORT` - O número da porta, entre 1024 e 65535, da porta na qual o host do banco de dados está escutando conexões de banco de dados.

- USER_NAME - O nome com o qual realizar login no banco de dados. O valor da string para USER_NAME é "USERNAME".
- PASSWORD - Uma senha, se for usada, para o nome do usuário.
- ENCRYPTED_PASSWORD - Ao habilitar a proteção por senha de conexão, definindo as configurações de criptografia ConnectionPasswordEncryption no Catálogo de dados, este campo armazena a senha criptografada.
- JDBC_DRIVER_JAR_URI – o caminho do Amazon Simple Storage Service (Amazon S3) do arquivo JAR que contém o driver JDBC a ser usado.
- JDBC_DRIVER_CLASS_NAME - A classe do nome do driver do JDBC a ser usado.
- JDBC_ENGINE - O nome do mecanismo JDBC a ser usado.
- JDBC_ENGINE_VERSION - A versão do nome do mecanismo JDBC a ser usado.
- CONFIG_FILES – (reservado para uso futuro.)
- INSTANCE_ID - O ID da instância a ser usada.
- JDBC_CONNECTION_URL – o URL para conexão com uma fonte de dados JDBC.
- JDBC_ENFORCE_SSL: uma string booliana (true, false) que especifica se a correspondência Secure Sockets Layer (SSL) com o hostname será aplicada para a conexão JDBC no cliente. O padrão é falso.
- CUSTOM_JDBC_CERT: um local do Amazon S3 que especifica o certificado raiz do cliente. O AWS Glue usa esse certificado raiz para validar o certificado do cliente ao se conectar ao banco de dados do cliente. O AWS Glue processa somente certificados X.509. O certificado fornecido deve ter codificação DER e ser fornecido no formato PEM de codificação Base64.
- SKIP_CUSTOM_JDBC_CERT_VALIDATION: Por padrão, é `false`. O AWS Glue valida o algoritmo de assinatura e o algoritmo de chave pública de requerente para o certificado do cliente. Os únicos algoritmos permitidos para o algoritmo de assinatura são SHA256withRSA, SHA384withRSA ou SHA512withRSA. Para o algoritmo de chave pública do assunto, o comprimento da chave deve ser pelo menos 2048. Você pode definir o valor dessa propriedade como `true` para ignorar a validação do certificado do cliente pelo AWS Glue.
- CUSTOM_JDBC_CERT_STRING — uma string de certificado JDBC personalizado que é usada para correspondência de domínio ou correspondência de nome distinto para evitar um ataque “man-in-the-middle”. No banco de dados Oracle, ela é usada como SSL_SERVER_CERT_DN; no Microsoft SQL Server, ela é usada como `hostNameInCertificate`.
- CONNECTION_URL – o URL para conexão com uma fonte de dados geral (não JDBC).
- SECRET_ID: o ID secreto usado para o gerente secreto de credenciais.

- `CONNECTOR_URL`: o URL do conector para uma conexão MARKETPLACE ou CUSTOM (personalizada).
- `CONNECTOR_TYPE`: o tipo do conector para uma conexão MARKETPLACE ou CUSTOM (personalizada).
- `CONNECTOR_CLASS_NAME`: o nome da classe do conector para uma conexão MARKETPLACE ou CUSTOM (personalizada).
- `KAFKA_BOOTSTRAP_SERVERS` – Uma lista separada por vírgulas de pares de host e porta que são os endereços dos agentes do Apache Kafka em um cluster do Kafka ao qual um cliente Kafka se conectará e se executará o próprio bootstrap.
- `KAFKA_SSL_ENABLED`: para habilitar ou desabilitar o SSL em uma conexão Apache Kafka. O valor padrão é “true” (verdadeiro).
- `KAFKA_CUSTOM_CERT`: o URL do Amazon S3 para o arquivo de certificado CA privado (formato.pem). O padrão é uma string vazia.
- `KAFKA_SKIP_CUSTOM_CERT_VALIDATION`: para ignorar ou não a validação do arquivo de certificado CA. O AWS Glue valida para três algoritmos: SHA256withRSA, SHA384withRSA e SHA512withRSA. O valor padrão é “false” (falso).
- `KAFKA_CLIENT_KEYSTORE`: a localização do Amazon S3 do arquivo de repositório de chaves do cliente para autenticação do lado do cliente Kafka (opcional).
- `KAFKA_CLIENT_KEYSTORE_PASSWORD`: a senha para acessar o repositório de chaves fornecido (opcional).
- `KAFKA_CLIENT_KEY_PASSWORD`: um repositório de chaves pode consistir em várias chaves, então essa é a senha para acessar a chave do cliente a ser usada com a chave do lado do servidor Kafka (opcional).
- `ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD`: a versão criptografada da senha do repositório de chaves do cliente Kafka (se o usuário tiver a configuração de senhas de criptografia do AWS Glue selecionada).
- `ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD`: a versão criptografada da senha da chave do cliente Kafka (se o usuário tiver a configuração de senhas de criptografia do AWS Glue selecionada).
- `KAFKA_SASL_MECHANISM`: `"SCRAM-SHA-512"`, `"GSSAPI"`, `"AWS_MSK_IAM"` ou `"PLAIN"`. Esses são os mecanismos SASL compatíveis.
- `KAFKA_SASL_PLAIN_USERNAME`: um nome de usuário em texto simples usado para autenticar com o mecanismo "PLAIN".

- `KAFKA_SASL_PLAIN_PASSWORD`: uma senha em texto simples usado para autenticar com o mecanismo "PLAIN".
- `ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD`: a versão criptografada da senha Kafka SASL PLAIN (se o usuário tiver a configuração de senhas de criptografia AWS Glue selecionada).
- `KAFKA_SASL_SCRAM_USERNAME` - Um nome de usuário em texto simples usado para autenticar com o mecanismo "SCRAM-SHA-512".
- `KAFKA_SASL_SCRAM_PASSWORD` - Uma senha de texto simples usada para autenticar com o mecanismo "SCRAM-SHA-512".
- `ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD` - A versão criptografada da senha Kafka SASL SCRAM (se o usuário tiver a configuração de senhas de criptografia AWS Glue selecionada).
- `KAFKA_SASL_SCRAM_SECRETS_ARN`: o nome do recurso da Amazon de um segredo do AWS Secrets Manager.
- `KAFKA_SASL_GSSAPI_KEYTAB` - A localização S3 de um arquivo Kerberos `keytab`. Um keytab armazena chaves de longo prazo para um ou mais principais. Para obter mais informações, consulte Documentação do MIT Kerberos: Keytab.
- `KAFKA_SASL_GSSAPI_KRB5_CONF` - A localização S3 de um arquivo Kerberos `krb5.conf`. Um krb5.conf armazena informações de configuração do Kerberos, como a localização do servidor KDC. Para obter mais informações, consulte Documentação do MIT Kerberos: krb5.conf.
- `KAFKA_SASL_GSSAPI_SERVICE` - O nome do serviço Kerberos, conforme definido com `sasl.kerberos.service.name` na sua Configuração do Kafka.
- `KAFKA_SASL_GSSAPI_PRINCIPAL`- O nome do princial Kerberos usado por AWS Glue. Para obter mais informações, consulte Documentação do Kafka: Configurando agentes do Kafka.

- `PhysicalConnectionRequirements` – Um objeto PhysicalConnectionRequirements.

  Os requisitos de conexão física, como nuvem virtual privada (VPC) e `SecurityGroup`, que são necessários para fazer essa conexão com êxito.

- `CreationTime` – Timestamp.

  A marca de data e hora em que essa definição de conexão foi criada.

- `LastUpdatedTime` – Timestamp.

  A marca de data e hora em que a definição de conexão foi atualizada pela última vez.

- `LastUpdatedBy` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O usuário, grupo ou função que atualizou pela última vez essa definição de conexão.

- `Status` – String UTF-8 (valores válidos: `READY` | `IN_PROGRESS` | `FAILED`).

  O status da conexão. Pode ser `READY`, `IN_PROGRESS` ou `FAILED`.

- `StatusReason`: string UTF-8 com não menos que 1 nem mais que 16.384 bytes de comprimento.

  O motivo do status da conexão.

- `LastConnectionValidationTime` – Timestamp.

  Uma marca de data e hora da última validação dessa conexão.

- `AuthenticationConfiguration` – Um objeto AuthenticationConfiguration.

  As propriedades de autenticação da conexão.

Estrutura ConnectionInput

Uma estrutura usada para especificar uma conexão para criar ou atualizar.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da conexão.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A descrição da conexão.

- `ConnectionType`: obrigatório: string UTF-8 (valores válidos: `JDBC` | `SFTP` | `MONGODB` | `KAFKA` | `NETWORK` | `MARKETPLACE` | `CUSTOM` | `SALESFORCE`).

  O tipo da conexão. No momento, estes tipos são compatíveis:

  - `JDBC` – Designa uma conexão com um banco de dados por meio do Java Database Connectivity (JDBC).

    As conexões de `JDBC` usam o seguinte ConnectionParameters.

    - Obrigatório: todos (`HOST`, `PORT`, `JDBC_ENGINE`) ou `JDBC_CONNECTION_URL`.

  - Obrigatório: todos (USERNAME, PASSWORD) ou SECRET_ID.
  - Opcional: JDBC_ENFORCE_SSL, CUSTOM_JDBC_CERT, CUSTOM_JDBC_CERT_STRING, SKIP_CUSTOM_JDBC_CERT_VALIDATION. Esses parâmetros são usados para configurar o SSL com o JDBC.
- KAFKA – Designa uma conexão com uma plataforma de streaming do Apache Kafka.

  As conexões de KAFKA usam o seguinte ConnectionParameters.

  - Obrigatório: KAFKA_BOOTSTRAP_SERVERS.
  - Opcional: KAFKA_SSL_ENABLED, KAFKA_CUSTOM_CERT, KAFKA_SKIP_CUSTOM_CERT_VALIDATION. Esses parâmetros são usados para configurar o SSL com o KAFKA.
  - Opcional: KAFKA_CLIENT_KEYSTORE, KAFKA_CLIENT_KEYSTORE_PASSWORD, KAFKA_CLIENT_KEY_PASSWORD, ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD, ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD. Esses parâmetros são usados para definir a configuração de cliente TLS com SSL no KAFKA.
  - Opcional: KAFKA_SASL_MECHANISM. Pode ser especificado como SCRAM-SHA-512, GSSAPI ou AWS_MSK_IAM.
  - Opcional: KAFKA_SASL_SCRAM_USERNAME, KAFKA_SASL_SCRAM_PASSWORD, ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD. Esses parâmetros são usados para configurar a autenticação SASL/SCRAM-SHA-512 com o KAFKA.
  - Opcional: KAFKA_SASL_GSSAPI_KEYTAB, KAFKA_SASL_GSSAPI_KRB5_CONF, KAFKA_SASL_GSSAPI_SERVICE, KAFKA_SASL_GSSAPI_PRINCIPAL. Esses parâmetros são usados para configurar a autenticação SASL/GSSAPI com o KAFKA.
- MONGODB – Designa uma conexão com um banco de dados de documentos do MongoDB.

  As conexões de MONGODB usam o seguinte ConnectionParameters.

  - Obrigatório: CONNECTION_URL.
  - Obrigatório: todos (USERNAME, PASSWORD) ou SECRET_ID.
- SALESFORCE: define uma conexão com o Salesforce usando a autenticação OAuth.
  - Requer que o membro AuthenticationConfiguration seja configurado.
- NETWORK: designa uma conexão de rede a uma fonte de dados dentro de um ambiente do Amazon Virtual Private Cloud (Amazon VPC).

As conexões de NETWORK não exigem ConnectionParameters. Em vez disso, forneça PhysicalConnectionRequirements.

- MARKETPLACE: usa as definições de configurações contidas em um conector adquirido de AWS Marketplace para ler e gravar em armazenamentos de dados que não são suportados nativamente pelo AWS Glue.

  As conexões de MARKETPLACE usam o seguinte ConnectionParameters.

  - Obrigatório: CONNECTOR_TYPE, CONNECTOR_URL, CONNECTOR_CLASS_NAME, CONNECTION_URL.
  - Obrigatório para conexões de JDBC CONNECTOR_TYPE: todos (USERNAME, PASSWORD) ou SECRET_ID.

- CUSTOM: usa as definições de configurações contidas em um conector personalizado para ler e gravar em armazenamentos de dados que não são suportados nativamente pelo AWS Glue.

Não há suporte ao SFTP.

Para obter mais informações sobre como ConnectionProperties opcionais são usadas para configurar recursos no AWS Glue, consulte Propriedades da conexão do AWS Glue.

Para obter mais informações sobre como ConnectionProperties opcionais são usadas para configurar recursos no AWS Glue Studio, consulte Usar conectores e conexões.

- MatchCriteria – Uma matriz de strings UTF-8, no máximo 10 strings.

  Uma lista de critérios que podem ser usados na seleção dessa conexão.

- ConnectionProperties: obrigatório: uma matriz de mapa dos pares de chave-valor, no máximo 100 pares.

  Cada chave é uma string 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 é um string de valor, com não superior a 1024 bytes.

Esses pares de chave/valor definem parâmetros para a conexão.

- `PhysicalConnectionRequirements` – Um objeto PhysicalConnectionRequirements.

  Os requisitos de conexão física, como nuvem privada virtual (VPC) e `SecurityGroup`, que são necessários para estabelecer essa conexão com êxito.

- `AuthenticationConfiguration` – Um objeto AuthenticationConfigurationInput.

  As propriedades de autenticação da conexão. Usadas para conexão com o Salesforce.

- `ValidateCredentials` – Booleano.

  Um sinalizador para validar as credenciais durante a criação da conexão. Usadas para conexão com o Salesforce. O padrão é true (verdadeiro).

Estrutura PhysicalConnectionRequirements

O aplicativo cliente OAuth na resposta GetConnection.

Campos

- `SubnetId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da sub-rede usada pela conexão.

- `SecurityGroupIdList` – Uma matriz de strings UTF-8, no máximo 50 strings.

  A lista de IDs de security group usada pela conexão.

- `AvailabilityZone` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  A zona de disponibilidade da conexão.

Estrutura GetConnectionsFilter

Filtra as definições de conexão retornadas pela `GetConnections` operação da API.

Campos

- `MatchCriteria` – Uma matriz de strings UTF-8, no máximo 10 strings.

  Uma cadeia de critérios que deve corresponder aos critérios registrados na definição de conexão para a definição de conexão a ser retornada.
- `ConnectionType` – String UTF-8 (valores válidos: `JDBC` | `SFTP` | `MONGODB` | `KAFKA` | `NETWORK` | `MARKETPLACE` | `CUSTOM` | `SALESFORCE`).

  O tipo de conexão a ser retornada. No momento, o SFTP não é compatível.

Operações

- Ação CreateConnection (Python: create_connection)
- Ação DeleteConnection (Python: delete_connection)
- Ação GetConnection (Python: get_connection)
- Ação GetConnections (Python: get_connections)
- Ação UpdateConnection (Python: update_connection)
- Ação BatchDeleteConnection (Python: batch_delete_connection)

Ação CreateConnection (Python: create_connection)

Cria uma definição de conexão no catálogo de dados.

As conexões usadas para criar recursos federados exigem a permissão `glue:PassConnection` do IAM.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que a conexão será criada. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.
- `ConnectionInput` – Obrigatório: um objeto ConnectionInput.

  Um objeto `ConnectionInput` que define a conexão a ser criada.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags que você atribui à conexão.

Resposta

- `CreateConnectionStatus` – String UTF-8 (valores válidos: `READY` | `IN_PROGRESS` | `FAILED`).

  O status da solicitação de criação da conexão. A solicitação pode levar algum tempo para determinados tipos de autenticação, por exemplo, ao criar uma conexão OAuth com troca de tokens via VPC.

Erros

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`

Ação DeleteConnection (Python: delete_connection)

Exclui uma conexão do catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que a conexão residirá. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.
- `ConnectionName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da conexão a ser excluída.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `OperationTimeoutException`

Ação GetConnection (Python: get_connection)

Recupera uma definição de conexão do catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que a conexão residirá. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.
- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de conexão a ser recuperada.
- `HidePassword` – Booleano.

Permite recuperar os metadados de conexão sem retornar a senha. Por exemplo, o console do AWS Glue usa esse sinalizador para recuperar a conexão e não exibe a senha. Defina esse parâmetro quando o autor da chamada não tem permissão para usar a chave do AWS KMS para descriptografar a senha, mas tem permissão para acessar o restante das propriedades da conexão.

Resposta

- `Connection` – Um objeto Conexão.

  A definição de conexão solicitada.

Erros

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

Ação GetConnections (Python: get_connections)

Recupera uma lista de definições de conexão do catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que as conexões residirão. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.
- `Filter` – Um objeto GetConnectionsFilter.

  Um filtro que controla quais conexões são retornadas.
- `HidePassword` – Booleano.

  Permite recuperar os metadados de conexão sem retornar a senha. Por exemplo, o console do AWS Glue usa esse sinalizador para recuperar a conexão e não exibe a senha. Defina esse

parâmetro quando o autor da chamada não tem permissão para usar a chave do AWS KMS para descriptografar a senha, mas tem permissão para acessar o restante das propriedades da conexão.

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de conexões a ser retornado em uma resposta.

Resposta

- `ConnectionList` – Uma matriz de objetos Conexão.

  Uma lista das definições de conexão solicitadas.

- `NextToken` – String UTF-8.

  Um token de continuação, se a lista de conexões retornada não incluir a última das conexões filtradas.

Erros

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

Ação UpdateConnection (Python: update_connection)

Atualiza uma definição de conexão no catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que a conexão residirá. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de conexão a ser atualizada.

- `ConnectionInput` – Obrigatório: um objeto ConnectionInput.

  Um objeto `ConnectionInput` que redefine a conexão em questão.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

Ação BatchDeleteConnection (Python: batch_delete_connection)

Exclui uma lista de definições de conexão do catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que as conexões residirão. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `ConnectionNameList` – Obrigatório: uma matriz de strings UTF-8, no máximo 25 strings.

  Uma lista de nomes das conexões a serem excluídas.

Resposta

- `Succeeded` – Uma matriz de strings UTF-8.

  Uma lista de nomes das definições de conexão que foram excluídas com sucesso.
- `Errors` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é um objeto ErrorDetail.

  Um mapa dos nomes das conexões que não foram excluídas com sucesso (para acessar detalhes do erro).

Erros

- `InternalServiceException`
- `OperationTimeoutException`

Configuração da autenticação

- Estrutura AuthenticationConfiguration
- Estrutura AuthenticationConfigurationInput
- Estrutura OAuth2Properties
- Estrutura OAuth2PropertiesInput
- Estrutura OAuth2ClientApplication
- Estrutura AuthorizationCodeProperties

Estrutura AuthenticationConfiguration

Uma estrutura que contém a configuração de autenticação.

Campos

- `AuthenticationType` – String UTF-8 (valores válidos: `BASIC` | `OAUTH2` | `CUSTOM`).

  Uma estrutura que contém a configuração de autenticação.

- `SecretArn` – String UTF-8 correspondente a Custom string pattern #11.

  O ARN do gerenciador de segredos para armazenar credenciais.

- `OAuth2Properties` – Um objeto OAuth2Properties.

  As propriedades da autenticação OAuth2.

Estrutura AuthenticationConfigurationInput

Uma estrutura que contém a configuração de autenticação na solicitação CreateConnection.

Campos

- `AuthenticationType` – String UTF-8 (valores válidos: `BASIC` | `OAUTH2` | `CUSTOM`).

  Uma estrutura que contém a configuração de autenticação na solicitação CreateConnection.

- `SecretArn` – String UTF-8 correspondente a Custom string pattern #11.

  O ARN do gerenciador de segredos para armazenar as credenciais na solicitação CreateConnection.

- `OAuth2Properties` – Um objeto OAuth2PropertiesInput.

  As propriedades da autenticação OAuth2 na solicitação CreateConnection.

Estrutura OAuth2Properties

Uma estrutura que contém propriedades da autenticação OAuth2.

Campos

- `OAuth2GrantType` – String UTF-8 (valores válidos: `AUTHORIZATION_CODE` | `CLIENT_CREDENTIALS` | `JWT_BEARER`).

  O tipo de concessão de OAuth2. Por exemplo, `AUTHORIZATION_CODE`, `JWT_BEARER` ou `CLIENT_CREDENTIALS`.

- `OAuth2ClientApplication` – Um objeto OAuth2ClientApplication.

  O tipo de aplicação do cliente. Por exemplo, AWS_MANAGED ou USER_MANAGED.

- `TokenUrl`: string UTF-8, com não mais que 256 bytes de comprimento, correspondente a Custom string pattern #12.

  O URL do servidor de autenticação do provedor para trocar um código de autorização por um token de acesso.
- `TokenUrlParametersMap` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 512 bytes.

  Um mapa de parâmetros que são adicionados à solicitação de token `GET`.

Estrutura OAuth2PropertiesInput

Uma estrutura que contém propriedades de OAuth2 na solicitação CreateConnection.

Campos

- `OAuth2GrantType` – String UTF-8 (valores válidos: `AUTHORIZATION_CODE` | `CLIENT_CREDENTIALS` | `JWT_BEARER`).

  O tipo de concessão OAuth2 na solicitação CreateConnection. Por exemplo, `AUTHORIZATION_CODE`, `JWT_BEARER` ou `CLIENT_CREDENTIALS`.
- `OAuth2ClientApplication` – Um objeto OAuth2ClientApplication.

  O tipo de aplicação cliente na solicitação CreateConnection. Por exemplo, o `AWS_MANAGED` ou o `USER_MANAGED`.
- `TokenUrl`: string UTF-8, com não mais que 256 bytes de comprimento, correspondente a Custom string pattern #12.

  O URL do servidor de autenticação do provedor para trocar um código de autorização por um token de acesso.
- `TokenUrlParametersMap` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 512 bytes.

  Um mapa de parâmetros que são adicionados à solicitação de token `GET`.

- `AuthorizationCodeProperties` – Um objeto AuthorizationCodeProperties.

  O conjunto de propriedades necessárias para o tipo de concessão `AUTHORIZATION_CODE` de OAuth2.

Estrutura OAuth2ClientApplication

A aplicação cliente OAuth2 usada para a conexão.

Campos

- `UserManagedClientApplicationClientId` – String UTF-8 com comprimento não superior a 2048 bytes, correspondente a Custom string pattern #13.

  O ClientID da aplicação cliente se ClientAppType for `USER_MANAGED`.
- `AWSManagedClientApplicationReference` – String UTF-8 com comprimento não superior a 2048 bytes, correspondente a Custom string pattern #13.

  A referência à aplicação cliente do lado SaaS que é gerenciada pela AWS.

Estrutura AuthorizationCodeProperties

O conjunto de propriedades necessárias para o fluxo de trabalho do tipo de concessão `AUTHORIZATION_CODE` de OAuth2.

Campos

- `AuthorizationCode`: string UTF-8, com pelo menos 1 e não mais de 4096 bytes de comprimento, correspondente a Custom string pattern #13.

  Um código de autorização a ser usado na terceira etapa do fluxo de trabalho de concessão de `AUTHORIZATION_CODE`. Esse é um código de uso único que se torna inválido quando trocado por um token de acesso, portanto, é aceitável ter esse valor como parâmetro de solicitação.
- `RedirectUri`: string UTF-8, com não mais que 512 bytes de comprimento, correspondente a Custom string pattern #14.

  O URI de redirecionamento para o qual o usuário é redirecionado pelo servidor de autorização ao emitir um código de autorização. O URI é usado posteriormente quando o código de autorização é trocado por um token de acesso.

API User-defined Function

A API User-defined Function descreve os tipos de dados e as operações do AWS Glue usados para trabalhar com funções.

Tipos de dados

- Estrutura UserDefinedFunction
- Estrutura UserDefinedFunctionInput

Estrutura UserDefinedFunction

Representa o equivalente a uma definição de função do Hive definida pelo usuário (UDF).

Campos

- `FunctionName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome da função.
- `DatabaseName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogo que contém a função.
- `ClassName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  A classe Java que contém o código da função.
- `OwnerName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O proprietário da função.
- `OwnerType` – String UTF-8 (valores válidos: `USER` | `ROLE` | `GROUP`).

  Tipo do proprietário.
- `CreateTime` – Timestamp.

  A hora em que a função foi criada.

- `ResourceUris` – Uma matriz de objetos ResourceUri, não mais de 1000 estruturas.

  Os URIs do recurso para a função.
- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do Data Catalog em que a função reside.

Estrutura UserDefinedFunctionInput

Uma estrutura usada para criar ou atualizar uma função definida pelo usuário.

Campos

- `FunctionName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome da função.
- `ClassName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  A classe Java que contém o código da função.
- `OwnerName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O proprietário da função.
- `OwnerType` – String UTF-8 (valores válidos: `USER` | `ROLE` | `GROUP`).

  Tipo do proprietário.
- `ResourceUris` – Uma matriz de objetos ResourceUri, não mais de 1000 estruturas.

  Os URIs do recurso para a função.

Operações

- Ação CreateUserDefinedFunction (Python: create_user_defined_function)
- Ação UpdateUserDefinedFunction (Python: update_user_defined_function)
- Ação DeleteUserDefinedFunction (Python: delete_user_defined_function)

- Ação GetUserDefinedFunction (Python: get_user_defined_function)
- Ação GetUserDefinedFunctions (Python: get_user_defined_functions)

Ação CreateUserDefinedFunction (Python: create_user_defined_function)

Cria uma definição de nova função no catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados em que a função será criada. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados de catálogos no qual a função será criada.
- `FunctionInput` – Obrigatório: um objeto UserDefinedFunctionInput.

  Um objeto `FunctionInput` que define a função a ser criada no catálogo de dados.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`

Ação UpdateUserDefinedFunction (Python: update_user_defined_function)

Atualiza uma definição de função existente no catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual está a função a ser atualizada. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados do catálogo no qual está a função a ser atualizada.

- `FunctionName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome da função.

- `FunctionInput` – Obrigatório: um objeto UserDefinedFunctionInput.

  Um objeto `FunctionInput` que redefine a função no Catálogo de dados.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Ação DeleteUserDefinedFunction (Python: delete_user_defined_function)

Exclui uma definição função existente do catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual está a função a ser excluída. Se nenhum valor for fornecido, o ID da conta da AWS será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados do catálogo no qual a função está.
- `FunctionName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de função a ser excluída.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetUserDefinedFunction (Python: get_user_defined_function)

Recupera uma definição de função especificada do catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual está a função a ser recuperada. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados do catálogo no qual a função está.
- `FunctionName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome da função.

Resposta

- `UserDefinedFunction` – Um objeto UserDefinedFunction.

  A definição de função solicitada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Ação GetUserDefinedFunctions (Python: get_user_defined_functions)

Recupera várias definições de função do Catálogo de dados.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O ID do catálogo de dados no qual estão as funções a serem recuperadas. Se nenhum for fornecido, o ID da conta da AWS será usado por padrão.

- `DatabaseName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados do catálogo no qual as funções estão localizadas. Se nenhum for fornecido, as funções de todos os bancos de dados no catálogo serão retornadas.

- `Pattern` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Uma string de padrão de função-nome opcional que filtra as definições de função retornadas.

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

- `MaxResults`: número (inteiro), não menos do que 1 ou superior a 100.

  O número máximo de funções a ser retornado em uma resposta.

Resposta

- `UserDefinedFunctions` – Uma matriz de objetos UserDefinedFunction.

  Uma lista das definições de função solicitadas.

- `NextToken` – String UTF-8.

  Um token de continuação, se a lista de funções retornadas não incluir a última função solicitada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`

Importar um catálogo do Athena para AWS Glue

A API Migration descreve os tipos de dados e as operações do AWS Glue relacionados à migração de um catálogo de dados do Athena para o AWS Glue.

Tipos de dados

- Estrutura CatalogImportStatus

Estrutura CatalogImportStatus

Uma estrutura que contém informações sobre o status da migração.

Campos

- `ImportCompleted` – Booleano.

  `True` se a migração for concluída, ou `False` caso contrário.
- `ImportTime` – Timestamp.

  A hora em que a migração foi iniciada.
- `ImportedBy` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da pessoa que iniciou a migração.

Operações

- Ação ImportCatalogToGlue (Python: import_catalog_to_glue)
- Ação GetCatalogImportStatus (Python: get_catalog_import_status)

Ação ImportCatalogToGlue (Python: import_catalog_to_glue)

Importa um Data Catalog existente do Amazon Athena para o AWS Glue.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O ID do catálogo a ser importado. No momento, deve ser o ID da conta da AWS.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InternalServiceException`
- `OperationTimeoutException`

Ação GetCatalogImportStatus (Python: get_catalog_import_status)

Recupera o status de uma operação de migração.

Solicitação

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo a ser migrado. No momento, deve ser o ID da conta da AWS.

Resposta

- `ImportStatus` – Um objeto CatalogImportStatus.

  O status de migração do catálogo especificado.

Erros

- `InternalServiceException`
- `OperationTimeoutException`

API do otimizador de tabelas

A API do otimizador de tabelas descreve a AWS Glue API para ativar a compactação para melhorar o desempenho de leitura.

Tipos de dados

- TableOptimizer estrutura
- TableOptimizerConfiguration estrutura
- TableOptimizerRun estrutura
- RunMetrics estrutura
- BatchGetTableOptimizerEntry estrutura
- BatchTableOptimizer estrutura
- BatchGetTableOptimizerError estrutura

TableOptimizer estrutura

Contém detalhes sobre um otimizador associado a uma tabela.

Campos

- `type` – String UTF-8 (valores válidos: `compaction="COMPACTION"`).

  O tipo de otimizador de tabelas. Atualmente, o único valor válido é `compaction`.
- `configuration` – Um objeto TableOptimizerConfiguration.

  Um objeto `TableOptimizerConfiguration` que foi especificado ao criar ou atualizar um otimizador de tabelas.
- `lastRun` – Um objeto TableOptimizerRun.

  Um objeto `TableOptimizerRun` que representa a última execução do otimizador de tabelas.

TableOptimizerConfiguration estrutura

Contém detalhes sobre a configuração de um otimizador de tabelas. Você passa essa configuração ao criar ou atualizar um otimizador de tabelas.

Campos

- `roleArn` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Single-line string pattern.

  Uma função passada pelo chamador que concede ao serviço permissão para atualizar os recursos associados ao otimizador em nome do chamador.
- `enabled` – Booleano.

  Se a otimização da tabela está habilitada.

TableOptimizerRun estrutura

Contém detalhes para a execução de um otimizador de tabelas.

Campos

- `eventType` – String UTF-8 (valores válidos: `starting="STARTING"` | `completed="COMPLETED"` | `failed="FAILED"` | `in_progress="IN_PROGRESS"`).

  Um tipo de evento que representa o status da execução do otimizador de tabelas.
- `startTimestamp` – Timestamp.

  Representa a marca de data e hora do epoch em que o trabalho de compactação foi iniciado no Lake Formation.
- `endTimestamp` – Timestamp.

  Representa a marca de data e hora do epoch em que o trabalho de compactação terminou.
- `metrics` – Um objeto RunMetrics.

  Um objeto `RunMetrics` que contém métricas para a execução do otimizador.
- `error` – String UTF-8.

  Um erro que ocorreu durante a execução do otimizador.

RunMetrics estrutura

Métricas para a execução do otimizador.

Campos

- `NumberOfBytesCompacted` – String UTF-8.

  O número de bytes removidos pela execução do trabalho de compactação.
- `NumberOfFilesCompacted` – String UTF-8.

  O número de arquivos removidos pela execução do trabalho de compactação.
- `NumberOfDpus` – String UTF-8.

  O número de horas de DPU consumidas pelo trabalho.
- `JobDurationInHour` – String UTF-8.

  A duração do trabalho, em horas.

BatchGetTableOptimizerEntry estrutura

Representa um otimizador de tabelas a ser recuperado na operação `BatchGetTableOptimizer`.

Campos

- `catalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo da tabela.
- `databaseName`: string UTF-8 com pelo menos 1 byte de comprimento.

  O nome do banco de dados no catálogo em que a tabela reside.
- `tableName`: string UTF-8 com pelo menos 1 byte de comprimento.

  O nome da tabela.
- `type` – String UTF-8 (valores válidos: `compaction="COMPACTION"`).

  O tipo de otimizador de tabelas.

BatchTableOptimizer estrutura

Contém detalhes de um dos otimizadores de tabela retornados pela operação `BatchGetTableOptimizer`.

Campos

- `catalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo da tabela.
- `databaseName`: string UTF-8 com pelo menos 1 byte de comprimento.

  O nome do banco de dados no catálogo em que a tabela reside.
- `tableName`: string UTF-8 com pelo menos 1 byte de comprimento.

  O nome da tabela.
- `tableOptimizer` – Um objeto TableOptimizer.

  Um objeto `TableOptimizer` que contém detalhes da configuração e da última execução de um otimizador de tabelas.

BatchGetTableOptimizerError estrutura

Contém detalhes sobre um dos erros na lista de erros retornada pela operação `BatchGetTableOptimizer`.

Campos

- `error` – Um objeto ErrorDetail.

  Um objeto `ErrorDetail` que contém detalhes de código e mensagens sobre o erro.
- `catalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo da tabela.
- `databaseName`: string UTF-8 com pelo menos 1 byte de comprimento.

  O nome do banco de dados no catálogo em que a tabela reside.
- `tableName`: string UTF-8 com pelo menos 1 byte de comprimento.

  O nome da tabela.
- `type` – String UTF-8 (valores válidos: `compaction="COMPACTION"`).

O tipo de otimizador de tabelas.

Operações

- GetTableOptimizer ação (Python: get_table_optimizer)
- BatchGetTableOptimizer ação (Python: batch_get_table_optimizer)
- ListTableOptimizerRuns ação (Python: list_table_optimizer_runs)
- CreateTableOptimizer ação (Python: create_table_optimizer)
- DeleteTableOptimizer ação (Python: delete_table_optimizer)
- UpdateTableOptimizer ação (Python: update_table_optimizer)

GetTableOptimizer ação (Python: get_table_optimizer)

Retorna a configuração de todos os otimizadores associados a uma tabela especificada.

Solicitação

- `CatalogId` – Obrigatório: string de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo da tabela.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no catálogo em que a tabela reside.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela.
- `Type`: obrigatório: string UTF-8 (valores válidos: `compaction="COMPACTION"`).

  O tipo de otimizador de tabelas.

Resposta

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo da tabela.

- `DatabaseName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no catálogo em que a tabela reside.

- `TableName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela.

- `TableOptimizer` – Um objeto TableOptimizer.

  O otimizador associado à tabela especificada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

BatchGetTableOptimizer ação (Python: batch_get_table_optimizer)

Retorna a configuração para os otimizadores de tabela especificados.

Solicitação

- `Entries` – Obrigatório: uma matriz de objetos BatchGetTableOptimizerEntry.

  Uma lista de objetos `BatchGetTableOptimizerEntry` que especifica os otimizadores de tabelas a serem recuperados.

Resposta

- `TableOptimizers` – Uma matriz de objetos BatchTableOptimizer.

  Uma lista dos objetos `BatchTableOptimizer`.
- `Failures` – Uma matriz de objetos BatchGetTableOptimizerError.

  Uma lista de erros da operação.

Erros

- `InternalServiceException`

ListTableOptimizerRuns ação (Python: list_table_optimizer_runs)

Lista o histórico de execuções anteriores do otimizador para uma tabela específica.

Solicitação

- `CatalogId` – Obrigatório: string de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo da tabela.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no catálogo em que a tabela reside.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela.
- `Type`: obrigatório: string UTF-8 (valores válidos: `compaction="COMPACTION"`).

  O tipo de otimizador de tabelas. Atualmente, o único valor válido é `compaction`.
- `MaxResults` – Número (íntegro).

  O número máximo de execuções do otimizador a serem retornadas em cada chamada.
- `NextToken` – String UTF-8.

Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `CatalogId` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo da tabela.

- `DatabaseName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no catálogo em que a tabela reside.

- `TableName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela.

- `NextToken` – String UTF-8.

  Um token de continuação para paginação da lista de execuções do otimizador retornada, retornado se o segmento atual da lista não for o último.

- `TableOptimizerRuns` – Uma matriz de objetos TableOptimizerRun.

  Uma lista das execuções de otimizador associadas a uma tabela.

Erros

- `EntityNotFoundException`
- `AccessDeniedException`
- `InvalidInputException`
- `InternalServiceException`

CreateTableOptimizer ação (Python: create_table_optimizer)

Cria um novo otimizador de tabela para uma função específica. No momento, `compaction` é o único tipo de otimizador que pode ser usado.

Solicitação

- `CatalogId` – Obrigatório: string de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo da tabela.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no catálogo em que a tabela reside.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela.

- `Type`: obrigatório: string UTF-8 (valores válidos: `compaction="COMPACTION"`).

  O tipo de otimizador de tabelas. Atualmente, o único valor válido é `compaction`.

- `TableOptimizerConfiguration` – Obrigatório: um objeto TableOptimizerConfiguration.

  Um `TableOptimizerConfiguration` objeto que representa a configuração de um otimizador de tabela.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `InternalServiceException`

DeleteTableOptimizer ação (Python: delete_table_optimizer)

Exclui um otimizador e todos os metadados associados de uma tabela. A otimização não será mais executada na tabela.

Solicitação

- `CatalogId` – Obrigatório: string de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo da tabela.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no catálogo em que a tabela reside.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela.
- `Type`: obrigatório: string UTF-8 (valores válidos: `compaction="COMPACTION"`).

  O tipo de otimizador de tabelas.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

UpdateTableOptimizer ação (Python: update_table_optimizer)

Atualiza a configuração de um otimizador de tabela existente.

Solicitação

- `CatalogId` – Obrigatório: string de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de catálogo da tabela.
- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no catálogo em que a tabela reside.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela.
- `Type`: obrigatório: string UTF-8 (valores válidos: `compaction="COMPACTION"`).

  O tipo de otimizador de tabelas. Atualmente, o único valor válido é `compaction`.
- `TableOptimizerConfiguration` – Obrigatório: um objeto TableOptimizerConfiguration.

  Um `TableOptimizerConfiguration` objeto que representa a configuração de um otimizador de tabela.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

API de crawlers e classificadores

A API Crawler and classifiers descreve os tipos de dados de crawler e de classificador do AWS Glue e inclui a API para criar, excluir, atualizar e listar crawlers ou classificadores.

Tópicos

API do classificador

A API de classificador descreve os tipos de dados de classificador do AWS Glue e inclui a API para criar, excluir, atualizar e listar classificadores.

Tipos de dados

Estrutura Classifier

Os classificadores são acionados durante um trabalho de rastreamento. Um classificador verifica se determinado arquivo está em um formato que pode ser processado. Se estiver, o classificador cria um esquema na forma de um objeto `StructType` que corresponde a esse formato de dados.

Você pode usar os classificadores padrão fornecidos pelo AWS Glue ou gravar seus próprios classificadores para categorizar melhor suas fontes de dados e especificar os esquemas apropriados a serem usados para eles. Ele pode ser um classificador `grok`, `XML` ou `JSON`, ou um `CSV` personalizado, conforme especificado em um dos campos no objeto `Classifier`.

Campos

- `GrokClassifier` – Um objeto GrokClassifier.

  Um classificador que usa `grok`.
- `XMLClassifier` – Um objeto XMLClassifier.

  Um classificador para o conteúdo XML.
- `JsonClassifier` – Um objeto JsonClassifier.

  Um classificador para o conteúdo JSON.
- `CsvClassifier` – Um objeto CsvClassifier.

  Um classificador para valores separados por vírgula (CSV, comma-separated values).

Estrutura GrokClassifier

Um classificador que usa padrões `grok`.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do classificador.
- `Classification` – Obrigatório: string UTF-8.

  Um identificador do formato de dados com o qual o classificador corresponde, como Twitter, JSON, logs da Omniture, e assim por diante.
- `CreationTime` – Timestamp.

  A hora em que o classificador foi registrado.
- `LastUpdated` – Timestamp.

A hora em que o classificador foi atualizado pela última vez.

- `Version` – Número (extenso).

  A versão do classificador.

- `GrokPattern` – Obrigatório: string UTF-8, superior a 1 e inferior a 2048 bytes de comprimento, correspondente a A Logstash Grok string pattern.

  O padrão grok aplicado a um armazenamento de dados por este classificador. Para obter mais informações, consulte os padrões integrados em Escrever classificadores personalizados.

- `CustomPatterns` – String UTF-8 com comprimento não superior a 16.000 bytes, correspondente a URI address multi-line string pattern.

  Padrões grok personalizados opcionais definidos por este classificador. Para obter mais informações, consulte os padrões personalizados em Escrever classificadores personalizados.

Estrutura XMLClassifier

Um classificador para o conteúdo `XML`.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do classificador.

- `Classification` – Obrigatório: string UTF-8.

  Um identificador do formato de dados aos quais o classificador corresponde.

- `CreationTime` – Timestamp.

  A hora em que o classificador foi registrado.

- `LastUpdated` – Timestamp.

  A hora em que o classificador foi atualizado pela última vez.

- `Version` – Número (extenso).

  A versão do classificador.

- `RowTag` – String UTF-8.

  A tag XML que designa o elemento que contém cada registro em um documento XML sendo analisado. Isso não pode identificar um elemento de fechamento automático (fechado por `/>`). Um elemento de linha vazio que contém somente atributos pode ser analisado desde que ele termine com uma tag de fechamento (por exemplo, `<row item_a="A" item_b="B"></row>` ok, mas `<row item_a="A" item_b="B" />` não).

Estrutura JsonClassifier

Um classificador para o conteúdo `JSON`.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do classificador.

- `CreationTime` – Timestamp.

  A hora em que o classificador foi registrado.

- `LastUpdated` – Timestamp.

  A hora em que o classificador foi atualizado pela última vez.

- `Version` – Número (extenso).

  A versão do classificador.

- `JsonPath` – Obrigatório: string UTF-8.

  Uma string `JsonPath` define os dados JSON para o classificador classificar. O AWS Glue oferece suporte a um subconjunto de JsonPath, conforme descrito em Gravar classificadores personalizados JsonPath.

Estrutura CsvClassifier

Um classificador para conteúdo `CSV` personalizado.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do classificador.

- `CreationTime` – Timestamp.

  A hora em que o classificador foi registrado.

- `LastUpdated` – Timestamp.

  A hora em que o classificador foi atualizado pela última vez.

- `Version` – Número (extenso).

  A versão do classificador.

- `Delimiter`: string UTF-8, não menos do que 1 ou superior a 1 byte de comprimento, correspondente a Custom string pattern #10.

  Um símbolo personalizado para indicar o que separa cada entrada de coluna na linha.

- `QuoteSymbol`: string UTF-8, não menos do que 1 ou superior a 1 byte de comprimento, correspondente a Custom string pattern #10.

  Um símbolo personalizado para indicar o que combina o conteúdo em um único valor da coluna. Deve ser diferente do delimitador de coluna.

- `ContainsHeader` – String UTF-8 (valores válidos: `UNKNOWN` | `PRESENT` | `ABSENT`).

  Indica se o arquivo CSV contém um cabeçalho.

- `Header` – Uma matriz de strings UTF-8.

  Uma lista de strings que representam nomes de coluna.

- `DisableValueTrimming` – Booleano.

  Especifica para não remover valores antes de identificar o tipo dos valores de coluna. O valor padrão é `true`.

- `AllowSingleColumn` – Booleano.

  Habilita o processamento de arquivos que contêm apenas uma coluna.

- `CustomDatatypeConfigured` – Booleano.

Permite que o tipo de dados personalizado seja configurado.

- `CustomDatatypes` – Uma matriz de strings UTF-8.

  Uma lista de tipos de dados personalizados, incluindo “BINARY”, “BOOLEAN”, “DATE”, “DECIMAL”, “DOUBLE”, “FLOAT”, “INT”, “LONG”, “SHORT”, “STRING”, “TIMESTAMP”.

- `Serde` – String UTF-8 (valores válidos: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  Define o SerDe para processamento de CSV no classificador, que será aplicado no catálogo de dados. Os valores válidos são `OpenCSVSerDe`, `LazySimpleSerDe` e `None`. Você pode especificar o valor `None` quando quiser que o crawler faça a detecção.

Estrutura CreateGrokClassifierRequest

Especifica um classificador `grok` a ser criado por `CreateClassifier`.

Campos

- `Classification` – Obrigatório: string UTF-8.

  Um identificador do formato de dados com o qual o classificador corresponde, como Twitter, JSON, logs da Omniture, Amazon CloudWatch Logs e assim por diante.

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do novo classificador.

- `GrokPattern` – Obrigatório: string UTF-8, superior a 1 e inferior a 2048 bytes de comprimento, correspondente a A Logstash Grok string pattern.

  O padrão grok que é usado por este classificador.

- `CustomPatterns` – String UTF-8 com comprimento não superior a 16.000 bytes, correspondente a URI address multi-line string pattern.

  Padrões grok personalizados opcionais usados por este classificador.

Estrutura UpdateGrokClassifierRequest

Especifica um classificador grok a ser atualizado quando transmitido para `UpdateClassifier`.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da `GrokClassifier`.
- `Classification` – String UTF-8.

  Um identificador do formato de dados com o qual o classificador corresponde, como Twitter, JSON, logs da Omniture, Amazon CloudWatch Logs e assim por diante.
- `GrokPattern` – String UTF-8, superior a 1 e inferior a 2048 bytes de comprimento, correspondente a A Logstash Grok string pattern.

  O padrão grok que é usado por este classificador.
- `CustomPatterns` – String UTF-8 com comprimento não superior a 16.000 bytes, correspondente a URI address multi-line string pattern.

  Padrões grok personalizados opcionais usados por este classificador.

Estrutura CreateXMLClassifierRequest

Especifica um classificador XML a ser criado por `CreateClassifier`.

Campos

- `Classification` – Obrigatório: string UTF-8.

  Um identificador do formato de dados aos quais o classificador corresponde.
- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do classificador.
- `RowTag` – String UTF-8.

  A tag XML que designa o elemento que contém cada registro em um documento XML sendo analisado. Isso não pode identificar um elemento de fechamento automático (fechado por `/>`). Um elemento de linha vazio que contém somente atributos pode ser analisado desde que ele termine com uma tag de fechamento (por exemplo, `<row item_a="A" item_b="B"></row>` ok, mas `<row item_a="A" item_b="B" />` não).

Estrutura UpdateXMLClassifierRequest

Especifica um classificador XML a ser atualizado.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do classificador.

- `Classification` – String UTF-8.

  Um identificador do formato de dados aos quais o classificador corresponde.

- `RowTag` – String UTF-8.

  A tag XML que designa o elemento que contém cada registro em um documento XML sendo analisado. Ela não pode identificar um elemento de fechamento automático (fechado por `/>`). Um elemento de linha vazio que contém somente atributos pode ser analisado desde que ele termine com uma tag de fechamento (por exemplo, `<row item_a="A" item_b="B"></row>` ok, mas `<row item_a="A" item_b="B" />` não).

Estrutura CreateJsonClassifierRequest

Especifica um classificador JSON a ser criado por `CreateClassifier`.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do classificador.

- `JsonPath` – Obrigatório: string UTF-8.

  Uma string `JsonPath` define os dados JSON para o classificador classificar. O AWS Glue oferece suporte a um subconjunto de JsonPath, conforme descrito em Gravar classificadores personalizados JsonPath.

Estrutura UpdateJsonClassifierRequest

Especifica um classificador JSON a ser atualizado.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do classificador.
- `JsonPath` – String UTF-8.

  Uma string `JsonPath` define os dados JSON para o classificador classificar. O AWS Glue oferece suporte a um subconjunto de JsonPath, conforme descrito em Gravar classificadores personalizados JsonPath.

Estrutura CreateCsvClassifierRequest

Especifica um classificador CSV personalizado a ser criado por `CreateClassifier`.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do classificador.
- `Delimiter`: string UTF-8, não menos do que 1 ou superior a 1 byte de comprimento, correspondente a Custom string pattern #10.

  Um símbolo personalizado para indicar o que separa cada entrada de coluna na linha.
- `QuoteSymbol`: string UTF-8, não menos do que 1 ou superior a 1 byte de comprimento, correspondente a Custom string pattern #10.

  Um símbolo personalizado para indicar o que combina o conteúdo em um único valor da coluna. Deve ser diferente do delimitador de coluna.
- `ContainsHeader` – String UTF-8 (valores válidos: `UNKNOWN` | `PRESENT` | `ABSENT`).

  Indica se o arquivo CSV contém um cabeçalho.
- `Header` – Uma matriz de strings UTF-8.

Uma lista de strings que representam nomes de coluna.

- `DisableValueTrimming` – Booleano.

  Especifica para não remover valores antes de identificar o tipo dos valores de coluna. O valor padrão é true.

- `AllowSingleColumn` – Booleano.

  Habilita o processamento de arquivos que contêm apenas uma coluna.

- `CustomDatatypeConfigured` – Booleano.

  Permite a configuração de tipos de dados personalizados.

- `CustomDatatypes` – Uma matriz de strings UTF-8.

  Cria uma lista de tipos de dados personalizados válidos.

- `Serde` – String UTF-8 (valores válidos: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  Define o SerDe para processamento de CSV no classificador, que será aplicado no catálogo de dados. Os valores válidos são `OpenCSVSerDe`, `LazySimpleSerDe` e `None`. Você pode especificar o valor `None` quando quiser que o crawler faça a detecção.

Estrutura UpdateCsvClassifierRequest

Especifica um classificador CSV personalizado a ser atualizado.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do classificador.

- `Delimiter`: string UTF-8, não menos do que 1 ou superior a 1 byte de comprimento, correspondente a Custom string pattern #10.

  Um símbolo personalizado para indicar o que separa cada entrada de coluna na linha.

- `QuoteSymbol`: string UTF-8, não menos do que 1 ou superior a 1 byte de comprimento, correspondente a Custom string pattern #10.

Um símbolo personalizado para indicar o que combina o conteúdo em um único valor da coluna. Deve ser diferente do delimitador de coluna.

- `ContainsHeader` – String UTF-8 (valores válidos: `UNKNOWN` | `PRESENT` | `ABSENT`).

  Indica se o arquivo CSV contém um cabeçalho.

- `Header` – Uma matriz de strings UTF-8.

  Uma lista de strings que representam nomes de coluna.

- `DisableValueTrimming` – Booleano.

  Especifica para não remover valores antes de identificar o tipo dos valores de coluna. O valor padrão é true.

- `AllowSingleColumn` – Booleano.

  Habilita o processamento de arquivos que contêm apenas uma coluna.

- `CustomDatatypeConfigured` – Booleano.

  Especifica a configuração de tipos de dados personalizados.

- `CustomDatatypes` – Uma matriz de strings UTF-8.

  Especifica uma lista de tipos de dados personalizados válidos.

- `Serde` – String UTF-8 (valores válidos: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  Define o SerDe para processamento de CSV no classificador, que será aplicado no catálogo de dados. Os valores válidos são `OpenCSVSerDe`, `LazySimpleSerDe` e `None`. Você pode especificar o valor `None` quando quiser que o crawler faça a detecção.

Operações

- Ação CreateClassifier (Python: create_classifier)
- Ação DeleteClassifier (Python: delete_classifier)
- Ação GetClassifier (Python: get_classifier)
- Ação GetClassifiers (Python: get_classifiers)
- Ação UpdateClassifier (Python: update_classifier)

Ação CreateClassifier (Python: create_classifier)

Cria um classificador na conta do usuário. Pode ser um `GrokClassifier`, um `XMLClassifier`, um `JsonClassifier` ou um `CsvClassifier`, dependendo de qual campo da solicitação está presente.

Solicitação

- `GrokClassifier` – Um objeto CreateGrokClassifierRequest.

  Um objeto `GrokClassifier` que especifica o classificador a ser criado.
- `XMLClassifier` – Um objeto CreateXMLClassifierRequest.

  Um objeto `XMLClassifier` que especifica o classificador a ser criado.
- `JsonClassifier` – Um objeto CreateJsonClassifierRequest.

  Um objeto `JsonClassifier` que especifica o classificador a ser criado.
- `CsvClassifier` – Um objeto CreateCsvClassifierRequest.

  Um objeto `CsvClassifier` que especifica o classificador a ser criado.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`

Ação DeleteClassifier (Python: delete_classifier)

Remove uma classificador do catálogo de dados.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

Nome do classificador a ser removido.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `OperationTimeoutException`

Ação GetClassifier (Python: get_classifier)

Recuperar um classificador por nome.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do classificador a ser recuperado.

Resposta

- `Classifier` – Um objeto Classificador.

  O classificador solicitado.

Erros

- `EntityNotFoundException`
- `OperationTimeoutException`

Ação GetClassifiers (Python: get_classifiers)

Lista todos os objetos de classificador no catálogo de dados.

Solicitação

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho da lista a ser retornada (opcional).
- `NextToken` – String UTF-8.

  Um token de continuação opcional.

Resposta

- `Classifiers` – Uma matriz de objetos Classificador.

  A lista solicitada de objetos do classificador.
- `NextToken` – String UTF-8.

  Um token de continuação.

Erros

- `OperationTimeoutException`

Ação UpdateClassifier (Python: update_classifier)

Modifica um classificador existente (`GrokClassifier`, `XMLClassifier`, `JsonClassifier` ou `CsvClassifier`, dependendo de qual campo estiver presente).

Solicitação

- `GrokClassifier` – Um objeto UpdateGrokClassifierRequest.

  Um objeto `GrokClassifier` com campos atualizados.
- `XMLClassifier` – Um objeto UpdateXMLClassifierRequest.

  Um objeto `XMLClassifier` com campos atualizados.
- `JsonClassifier` – Um objeto UpdateJsonClassifierRequest.

  Um objeto `JsonClassifier` com campos atualizados.
- `CsvClassifier` – Um objeto UpdateCsvClassifierRequest.

Um objeto `CsvClassifier` com campos atualizados.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InvalidInputException`
- `VersionMismatchException`
- `EntityNotFoundException`
- `OperationTimeoutException`

API do crawler

A API Crawler descreve os tipos de dados do AWS Glue rastreador, junto com a API para criar, excluir, atualizar e listar rastreadores.

Tipos de dados

- Estrutura Crawler
- Estrutura Schedule
- CrawlerTargets estrutura
- Estrutura S3Target
- Estrutura S3 DeltaCatalogTarget
- Estrutura S3 DeltaDirectTarget
- JdbcTarget estrutura
- Estrutura MongoDBTarget
- Estrutura DynamoDBTarget
- DeltaTarget estrutura
- IcebergTarget estrutura
- HudiTarget estrutura
- CatalogTarget estrutura

- CrawlerMetrics estrutura
- CrawlerHistory estrutura
- CrawlsFilter estrutura
- SchemaChangePolicy estrutura
- LastCrawlInfo estrutura
- RecrawlPolicy estrutura
- LineageConfiguration estrutura
- LakeFormationConfiguration estrutura

Estrutura Crawler

Especifica um programa de crawler que examina uma fonte de dados e usa classificadores para tentar determinar seu esquema. Se for bem-sucedido, o crawler registrará metadados da fonte de dados no AWS Glue Data Catalog.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do crawler.
- `Role` – String UTF-8.

  O nome de recurso da Amazon (ARN) de uma função do IAM que é usada para acessar os recursos do cliente, como o Amazon Simple Storage Service (Amazon S3).
- `Targets` – Um objeto CrawlerTargets.

  Uma coleção de destinos a serem rastreados.
- `DatabaseName` – String UTF-8.

  O nome do banco de dados no qual a saída do crawler é armazenada.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do crawler.
- `Classifiers` – Uma matriz de strings UTF-8.

Uma lista de strings UTF-8 que especificam os classificadores personalizados associados ao crawler.

- `RecrawlPolicy` – Um objeto RecrawlPolicy.

  Uma política que especifica se deseja rastrear todo o conjunto de dados novamente ou rastrear somente pastas que foram adicionadas desde a última execução do crawler.

- `SchemaChangePolicy` – Um objeto SchemaChangePolicy.

  A política que especifica os comportamentos de atualização e exclusão do crawler.

- `LineageConfiguration` – Um objeto LineageConfiguration.

  Uma configuração que especifica se a linhagem de dados está habilitada para o crawler.

- `State` – String UTF-8 (valores válidos: `READY` | `RUNNING` | `STOPPING`).

  Indica se o crawler está em execução ou se uma execução está pendente.

- `TablePrefix` – String UTF-8 com comprimento não superior a 128 bytes.

  O prefixo adicionado aos nomes das tabelas criadas.

- `Schedule` – Um objeto Schedule.

  Para crawlers programados, a programação quando o crawler é executado.

- `CrawlElapsedTime` – Número (extenso).

  Se o crawler estiver em execução, conterá o tempo decorrido total desde o início do rastreamento.

- `CreationTime` – Timestamp.

  A hora em que o crawler foi criado.

- `LastUpdated` – Timestamp.

  A hora em que o crawler foi atualizado pela última vez.

- `LastCrawl` – Um objeto LastCrawlInfo.

  O status do último rastreamento e informações de erro (se houver algum).

- `Version` – Número (extenso).

  A versão do crawler.

- `Configuration` – String UTF-8.

Informações de configuração do crawler. Esta string JSON com versionamento permite que os usuários especifiquem os aspectos do comportamento de um crawler. Para obter mais informações, consulte Definir opções de configuração do crawler.

- `CrawlerSecurityConfiguration` – String UTF-8 com comprimento não superior a 128 bytes.

  O nome da estrutura `SecurityConfiguration` a ser usada por este crawler.

- `LakeFormationConfiguration` – Um objeto LakeFormationConfiguration.

  Especifica se o rastreador deve usar AWS Lake Formation credenciais para o rastreador em vez das credenciais da função do IAM.

Estrutura Schedule

Um objeto de programação que usa uma instrução `cron` para programar um evento.

Campos

- `ScheduleExpression` – String UTF-8.

  Uma expressão `cron` usada para especificar a programação (consulte Programações baseadas em hora para tarefas e crawlers. Por exemplo, para executar algo todos os dias às 12h15 UTC, especifique: `cron(15 12 * * ? *)`.

- `State` – String UTF-8 (valores válidos: `SCHEDULED` | `NOT_SCHEDULED` | `TRANSITIONING`).

  O estado da programação.

CrawlerTargets estrutura

Especifica os armazenamentos de dados a serem rastreados.

Campos

- `S3Targets` – Uma matriz de objetos S3Target.

  Especifica os destinos do Amazon Simple Storage Service (Amazon S3).

- `JdbcTargets` – Uma matriz de objetos JdbcTarget.

  Especifica destinos JDBC.

- `MongoDBTargets` – Uma matriz de objetos MongoDBTarget.

  Especifica destinos do Amazon DocumentDB ou MongoDB.

- `DynamoDBTargets` – Uma matriz de objetos DynamoDBTarget.

  Especifica os destinos do Amazon DynamoDB.

- `CatalogTargets` – Uma matriz de objetos CatalogTarget.

  Especifica os AWS Glue Data Catalog alvos.

- `DeltaTargets` – Uma matriz de objetos DeltaTarget.

  Especifica os destinos do armazenamento de dados Delta.

- `IcebergTargets` – Uma matriz de objetos IcebergTarget.

  Especifica os destinos do armazenamento de dados do Apache Iceberg.

- `HudiTargets` – Uma matriz de objetos HudiTarget.

  Especifica os destinos do armazenamento de dados do Apache Hudi.

Estrutura S3Target

Especifica um armazenamento de dados no Amazon Simple Storage Service (Amazon S3).

Campos

- `Path` – String UTF-8.

  O caminho do destino do Amazon S3.

- `Exclusions` – Uma matriz de strings UTF-8.

  Uma lista de padrões glob utilizados para a exclusão do rastreamento. Para obter mais informações, consulte Catalogar tabelas com um crawler.

- `ConnectionName` – String UTF-8.

  O nome de uma conexão que permite que um trabalho ou crawler acesse dados no Amazon S3 em um ambiente do Amazon Virtual Private Cloud (Amazon VPC).

- `SampleSize` – Número (íntegro).

Define o número de arquivos em cada pasta de folha a serem rastreados ao realizar crawling de arquivos de amostra em um conjunto de dados. Se não for definido, todos os arquivos serão rastreados. Um valor válido é um número inteiro entre 1 e 249.

- `EventQueueArn` – String UTF-8.

  Um ARN válido do Amazon SQS. Por exemplo, `arn:aws:sqs:region:account:sqs`.

- `DlqEventQueueArn` – String UTF-8.

  Um ARN do SQS de mensagem morta válida da Amazon. Por exemplo, `arn:aws:sqs:region:account:deadLetterQueue`.

Estrutura S3 DeltaCatalogTarget

Especifica um destino que grava em uma fonte de dados do Delta Lake no Catálogo AWS Glue de Dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `PartitionKeys` – Uma matriz de strings UTF-8.

  Especifica o particionamento nativo usando uma sequência de chaves.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados na qual gravar.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados no qual gravar.

- `AdditionalOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

Especifica as opções de conexão adicionais para o conector.

- `SchemaChangePolicy` – Um objeto CatalogSchemaChangePolicy.

  Uma política que especifica o comportamentos de atualização do crawler.

Estrutura S3 DeltaDirectTarget

Especifica um destino que grava em uma fonte de dados do Delta Lake em Amazon S3.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `PartitionKeys` – Uma matriz de strings UTF-8.

  Especifica o particionamento nativo usando uma sequência de chaves.

- `Path` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O caminho do Amazon S3 da fonte de dados do Delta Lake na qual gravar.

- `Compression` – Obrigatório: string UTF-8 (valores válidos: `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  Especifica como os dados são compactados. Isso geralmente não é necessário se os dados tem uma extensão de arquivo padrão. Os possíveis valores são `"gzip"` e `"bzip"`).

- `Format`: obrigatório: string UTF-8 (valores válidos: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

  Especifica o formato de saída de dados para o destino.

- `AdditionalOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

Especifica as opções de conexão adicionais para o conector.

- `SchemaChangePolicy` – Um objeto DirectSchemaChangePolicy.

  Uma política que especifica o comportamentos de atualização do crawler.

JdbcTarget estrutura

Especifica um armazenamento de dados JDBC a ser rastreado.

Campos

- `ConnectionName` – String UTF-8.

  O nome da conexão a ser usada para se conectar com o destino JDBC.
- `Path` – String UTF-8.

  O caminho do destino do JDBC.
- `Exclusions` – Uma matriz de strings UTF-8.

  Uma lista de padrões glob utilizados para a exclusão do rastreamento. Para obter mais informações, consulte Catalogar tabelas com um crawler.
- `EnableAdditionalMetadata` – Uma matriz de strings UTF-8.

  Especifique um valor de `RAWTYPES` ou `COMMENTS` para habilitar metadados adicionais nas respostas da tabela. O `RAWTYPES` fornece o tipo de dados de nível nativo. O `COMMENTS` fornece comentários associados a uma coluna ou tabela no banco de dados.

  Se você não precisar de metadados adicionais, deixe o campo vazio.

Estrutura MongoDBTarget

Especifica um armazenamento de dados do Amazon DocumentDB ou MongoDB a ser rastreado.

Campos

- `ConnectionName` – String UTF-8.

  O nome da conexão a ser usada para se conectar com o destino do Amazon DocumentDB ou MongoDB.

- `Path` – String UTF-8.

  O caminho do destino do Amazon DocumentDB ou MongoDB (banco de dados/coleção).

- `ScanAll` – Booleano.

  Indica se deseja verificar todos os registros ou amostras de linhas da tabela. A verificação de todos os registros pode levar muito tempo quando a tabela não é de throughput alto.

  Um valor de `true` indica para verificar todos os registros, enquanto um valor de `false` indica para criar amostra dos registros. Se nenhum valor for especificado, o valor `true` será assumido como padrão.

Estrutura DynamoDBTarget

Especifica uma tabela do Amazon DynamoDB para ser rastreada.

Campos

- `Path` – String UTF-8.

  O nome da tabela do DynamoDB a ser rastreada.

- `scanAll` – Booleano.

  Indica se deseja verificar todos os registros ou amostras de linhas da tabela. A verificação de todos os registros pode levar muito tempo quando a tabela não é de throughput alto.

  Um valor de `true` indica para verificar todos os registros, enquanto um valor de `false` indica para criar amostra dos registros. Se nenhum valor for especificado, o valor `true` será assumido como padrão.

- `scanRate` – Número (duplo).

  A porcentagem das unidades de capacidade de leitura configuradas a serem usadas pelo AWS Glue rastreador. Unidades de capacidade de leitura é um termo definido pelo DynamoDB e é um valor numérico que atua como limitador de taxa para o número de leituras que podem ser executadas nessa tabela por segundo.

  Os valores válidos são nulos ou um valor entre 0,1 e 1,5. Um valor nulo é usado quando o usuário não fornece um valor e é usado como padrão 0,5 da unidade de capacidade de leitura configurada

(para tabelas provisionadas) ou 0,25 da unidade de capacidade de leitura máxima configurada (para tabelas que usam o modo sob demanda).

DeltaTarget estrutura

Especifica um armazenamento de dados Delta para rastrear uma ou mais tabelas Delta.

Campos

- `DeltaTables` – Uma matriz de strings UTF-8.

  Uma lista de caminhos do Amazon S3 para as tabelas Delta.
- `ConnectionName` – String UTF-8.

  O nome da conexão a ser usada para se conectar ao destino da tabela Delta.
- `WriteManifest` – Booleano.

  Especifica se os arquivos de manifesto devem ser gravados no caminho da tabela Delta.
- `CreateNativeDeltaTable` – Booleano.

  Especifica se o crawler criará tabelas nativas para permitir a integração com mecanismos de consulta compatíveis consulta direta ao log de transações do Delta.

IcebergTarget estrutura

Especifica uma fonte de dados do Apache Iceberg na qual as tabelas do Iceberg são armazenadas no Amazon S3.

Campos

- `Paths` – Uma matriz de strings UTF-8.

  Um ou mais Amazon S3 caminhos que contêm pastas de metadados do Iceberg como. `s3://bucket/prefix`
- `ConnectionName` – String UTF-8.

  O nome da conexão a ser usada para se conectar com o destino do Iceberg.
- `Exclusions` – Uma matriz de strings UTF-8.

Uma lista de padrões glob utilizados para a exclusão do rastreamento. Para obter mais informações, consulte Catalogar tabelas com um crawler.

- `MaximumTraversalDepth` – Número (íntegro).

  A profundidade máxima de Amazon S3 caminhos que o rastreador pode percorrer para descobrir a pasta de metadados do Iceberg em seu caminho. Amazon S3 Usado para limitar o runtime do crawler.

HudiTarget estrutura

Especifica uma fonte de dados do Apache Hudi.

Campos

- `Paths` – Uma matriz de strings UTF-8.

  Uma matriz de cadeias de caracteres de Amazon S3 localização para Hudi, cada uma indicando a pasta raiz na qual residem os arquivos de metadados de uma tabela Hudi. A pasta do Hudi pode estar localizada em uma pasta secundária da pasta raiz.

  O crawler examinará todas as pastas abaixo de um caminho para uma pasta do Hudi.
- `ConnectionName` – String UTF-8.

  O nome da conexão a ser usada para se conectar com o destino do Hudi. Se seus arquivos do Hudi estiverem armazenados em buckets que exigem autorização de VPC, você pode definir suas propriedades de conexão aqui.
- `Exclusions` – Uma matriz de strings UTF-8.

  Uma lista de padrões glob utilizados para a exclusão do rastreamento. Para obter mais informações, consulte Catalogar tabelas com um crawler.
- `MaximumTraversalDepth` – Número (íntegro).

  A profundidade máxima de Amazon S3 caminhos que o rastreador pode percorrer para descobrir a pasta de metadados Hudi em seu caminho. Amazon S3 Usado para limitar o runtime do crawler.

CatalogTarget estrutura

Especifica um AWS Glue Data Catalog alvo.

Campos

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados a ser sincronizado.

- `Tables` – Obrigatório: uma matriz de strings UTF-8, pelo menos 1 string.

  Uma lista de tabelas a serem sincronizadas.

- `ConnectionName` – String UTF-8.

  O nome da conexão de uma tabela de catálogo de dados baseada no Amazon S3 é um destino do crawl quando um tipo de conexão `Catalog` pareado a um tipo de conexão `NETWORK` é usado.

- `EventQueueArn` – String UTF-8.

  Um ARN válido do Amazon SQS. Por exemplo, `arn:aws:sqs:region:account:sqs`.

- `DlqEventQueueArn` – String UTF-8.

  Um ARN do SQS de mensagem morta válida da Amazon. Por exemplo, `arn:aws:sqs:region:account:deadLetterQueue`.

CrawlerMetrics estrutura

Métricas para um crawler especificado.

Campos

- `CrawlerName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do crawler.

- `TimeLeftSeconds` – Número (dobro), não mais do que None (Nenhum).

  O tempo estimado restante para concluir um rastreamento em execução.

- `StillEstimating` – Booleano.

  True se o crawler ainda estiver estimando quanto tempo demorará para concluir a execução.

- `LastRuntimeSeconds` – Número (dobro), não mais do que None (Nenhum).

A duração da execução mais recente do crawler em segundos.

- `MedianRuntimeSeconds` – Número (dobro), não mais do que None (Nenhum).

  A duração média da execução do crawler em segundos.

- `TablesCreated` – Número (inteiro), não mais do que None (Nenhum).

  O número de tabelas criadas por este crawler.

- `TablesUpdated` – Número (inteiro), não mais do que None (Nenhum).

  O número de tabelas atualizadas por este crawler.

- `TablesDeleted` – Número (inteiro), não mais do que None (Nenhum).

  O número de tabelas excluídas por este crawler.

CrawlerHistory estrutura

Contém as informações sobre uma execução de um crawler.

Campos

- `CrawlId` – String UTF-8.

  Um identificador UUID para cada crawl.

- `State` – String UTF-8 (valores válidos: `RUNNING` | `COMPLETED` | `FAILED` | `STOPPED`).

  O estado do crawl.

- `StartTime` – Timestamp.

  A data e a hora em que o monitoramento foi iniciado.

- `EndTime` – Timestamp.

  A data e a hora em que o crawl terminou.

- `Summary` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um resumo da execução para o crawl específico em JSON. Contém as partições e tabelas de catálogo que foram adicionadas, atualizadas ou excluídas.

- `ErrorMessage` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Em caso de erro, a mensagem de erro associada ao crawl.

- `LogGroup` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Log group string pattern.

  O grupo de logs associado ao crawl.

- `LogStream` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Log-stream string pattern.

  O fluxo de logs associado ao rastreamento.

- `MessagePrefix` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O prefixo de uma CloudWatch mensagem sobre esse rastreamento.

- `DPUHour` – Número (dobro), não mais do que None (Nenhum).

  O número de data processing units (DPU – Unidades de processamento de dados) usadas para o crawl em horas.

CrawlsFilter estrutura

Uma lista de campos, comparadores e valores que você pode usar para filtrar as execuções de crawler para um crawler especificado.

Campos

- `FieldName` – String UTF-8 (valores válidos: `CRAWL_ID` | `STATE` | `START_TIME` | `END_TIME` | `DPU_HOUR`).

  Uma chave usada para filtrar as execuções de crawler para um crawler especificado. Os valores válidos para cada um dos nomes de campo são:

  - `CRAWL_ID`: uma string que representa o identificador UUID para um crawl.
  - `STATE`: uma string que representa o estado do crawl.
  - `START_TIME` e `END_TIME`: o carimbo de data e hora de época em milissegundos.
  - `DPU_HOUR`: O número de horas de DPU usadas para o crawl.

- `FilterOperator` – String UTF-8 (valores válidos: `GT` | `GE` | `LT` | `LE` | `EQ` | `NE`).

  Um comparador definido que opera no valor. Os operadores disponíveis são:

  - `GT`: maior que.
  - `GE`: maior ou igual a.
  - `LT`: menor que.
  - `LE`: menor ou igual a.
  - `EQ`: igual a.
  - `NE`: não é igual a.
- `FieldValue` – String UTF-8.

  O valor fornecido para comparação no campo de crawl.

SchemaChangePolicy estrutura

Uma política que especifica os comportamentos de atualização e exclusão do crawler.

Campos

- `UpdateBehavior`: string UTF-8 (valores válidos: `LOG` | `UPDATE_IN_DATABASE`).

  O comportamento de atualização quando o crawler encontra um esquema alterado.
- `DeleteBehavior` – String UTF-8 (valores válidos: `LOG` | `DELETE_FROM_DATABASE` | `DEPRECATE_IN_DATABASE`).

  O comportamento de exclusão quando o crawler encontra um objeto excluído.

LastCrawlInfo estrutura

Informações de status e erro do rastreamento mais recente.

Campos

- `Status` – String UTF-8 (valores válidos: `SUCCEEDED` | `CANCELLED` | `FAILED`).

  Status do último rastreamento.
- `ErrorMessage` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

Se ocorrer um erro, as informações de erro do último rastreamento.

- `LogGroup` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Log group string pattern.

  O grupo de logs do último rastreamento.

- `LogStream` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Log-stream string pattern.

  O stream de logs do último rastreamento.

- `MessagePrefix` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O prefixo de uma mensagem sobre este rastreamento.

- `StartTime` – Timestamp.

  A hora em que o rastreamento foi iniciado.

RecrawlPolicy estrutura

Ao realizar o crawling de uma fonte de dados do Amazon S3 após a conclusão do primeiro rastreamento, especifica se deseja rastrear todo o conjunto de dados novamente ou somente pastas que foram adicionadas desde a última execução do crawler. Para obter mais informações, consulte Crawls incrementais no AWS Glue no guia do desenvolvedor.

Campos

- `RecrawlBehavior` – String UTF-8 (valores válidos: `CRAWL_EVERYTHING` | `CRAWL_NEW_FOLDERS_ONLY` | `CRAWL_EVENT_MODE`).

  Especifica se deseja rastrear todo o conjunto de dados novamente ou somente as pastas que foram adicionadas desde a última execução do crawler.

  Um valor de `CRAWL_EVERYTHING` especifica o crawling de todo o conjunto de dados novamente.

  Um valor de `CRAWL_NEW_FOLDERS_ONLY` especifica o crawling de somente as pastas que foram adicionadas desde a última execução do crawler.

Um valor de `CRAWL_EVENT_MODE` especifica o crawling somente das alterações identificadas pelos eventos do Amazon S3.

LineageConfiguration estrutura

Especifica as definições de configuração de linhagem de dados para o crawler.

Campos

- `CrawlerLineageSettings`: string UTF-8 (valores válidos: `ENABLE` | `DISABLE`).

  Especifica se a linhagem de dados está habilitada para o crawler. Os valores válidos são:

  - ENABLE (Habilitar): habilita a linhagem de dados para o crawler
  - DISABLE (Desabilitar): desabilita a linhagem de dados para o crawler

LakeFormationConfiguration estrutura

Especifica AWS Lake Formation as configurações do rastreador.

Campos

- `UseLakeFormationCredentials` – Booleano.

  Especifica se as AWS Lake Formation credenciais do rastreador devem ser usadas em vez das credenciais da função do IAM.

- `AccountId` – String UTF-8, não mais de 12 bytes.

  Obrigatório para rastreamentos de conta cruzada. Para os mesmos rastreamentos de conta que os dados de destino, isso pode ser deixado como nulo.

Operações

- CreateCrawler ação (Python: create_crawler)
- DeleteCrawler ação (Python: delete_crawler)
- GetCrawler ação (Python: get_crawler)
- GetCrawlers ação (Python: get_crawlers)

- GetCrawlerMetrics ação (Python: get_crawler_metrics)
- UpdateCrawler ação (Python: update_crawler)
- StartCrawler ação (Python: start_crawler)
- StopCrawler ação (Python: stop_crawler)
- BatchGetCrawlers ação (Python: batch_get_crawlers)
- ListCrawlers ação (Python: list_crawlers)
- ListCrawls ação (Python: list_crawls)

CreateCrawler ação (Python: create_crawler)

Cria um novo crawler com destinos específicos, função, configuração e programação opcional. Pelo menos um destino de rastreamento deve ser especificado no campos `s3Targets`, `jdbcTargets` ou `DynamoDBTargets`.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do novo crawler.
- `Role` – Obrigatório: string UTF-8.

  A função do IAM ou o nome de recurso da Amazon (ARN) de uma função do IAM usada pelo novo crawler para acessar os recursos do cliente.
- `DatabaseName` – String UTF-8.

  O AWS Glue banco de dados onde os resultados são gravados, tal como:`arn:aws:daylight:us-east-1::database/sometable/*`.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do novo crawler.
- `Targets` – Obrigatório: um objeto CrawlerTargets.

  Uma lista da coleção de destinos a serem rastreados.
- `Schedule` – String UTF-8.

Uma expressão `cron` usada para especificar a programação (consulte Programações baseadas em hora para tarefas e crawlers. Por exemplo, para executar algo todos os dias às 12h15 UTC, especifique: `cron(15 12 * * ? *)`.

- `Classifiers` – Uma matriz de strings UTF-8.

  Uma lista de classificadores personalizados que o usuário registrou. Por padrão, todos os classificadores integrados são incluídos em um rastreamento. No entanto, esses classificadores personalizados sempre substituem os classificadores padrão de uma determinada classificação.

- `TablePrefix` – String UTF-8 com comprimento não superior a 128 bytes.

  O prefixo que é usado para tabelas de catálogo criadas.

- `SchemaChangePolicy` – Um objeto SchemaChangePolicy.

  A política do comportamento de atualização e exclusão do crawler.

- `RecrawlPolicy` – Um objeto RecrawlPolicy.

  Uma política que especifica se deseja rastrear todo o conjunto de dados novamente ou rastrear somente pastas que foram adicionadas desde a última execução do crawler.

- `LineageConfiguration` – Um objeto LineageConfiguration.

  Especifica as definições de configuração de linhagem de dados para o crawler.

- `LakeFormationConfiguration` – Um objeto LakeFormationConfiguration.

  Especifica AWS Lake Formation as configurações do rastreador.

- `Configuration` – String UTF-8.

  Informações de configuração do crawler. Esta string JSON com versionamento permite que os usuários especifiquem os aspectos do comportamento de um crawler. Para obter mais informações, consulte Definir opções de configuração do crawler.

- `CrawlerSecurityConfiguration` – String UTF-8 com comprimento não superior a 128 bytes.

  O nome da estrutura `SecurityConfiguration` a ser usada por este crawler.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

As tags a serem usadas com essa solicitação de crawler. Você pode usar tags para limitar o acesso ao crawler. Para obter mais informações sobre tags em AWS Glue, consulte AWS Tags AWS Glue in no guia do desenvolvedor.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InvalidInputException`
- `AlreadyExistsException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

DeleteCrawler ação (Python: delete_crawler)

Remove um rastreador especificado do AWS Glue Data Catalog, a menos que o estado do rastreador seja. `RUNNING`

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do crawler a ser removido.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `CrawlerRunningException`
- `SchedulerTransitioningException`

- `OperationTimeoutException`

GetCrawler ação (Python: get_crawler)

Recupera metadados para um crawler especificado.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do crawler para o qual recuperar os metadados.

Resposta

- `Crawler` – Um objeto Crawler.

  Os metadados para o crawler especificado.

Erros

- `EntityNotFoundException`
- `OperationTimeoutException`

GetCrawlers ação (Python: get_crawlers)

Recupera metadados para todos os crawlers definidos na conta do cliente.

Solicitação

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número de crawlers a ser retornado em cada chamada.
- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma solicitação de continuação.

Resposta

- `Crawlers` – Uma matriz de objetos Crawler.

  Uma lista de metadados do crawler.

- `NextToken` – String UTF-8.

  Um token de continuação, se a lista retornada não tiver chegado ao final conforme definido nesta conta de cliente.

Erros

- `OperationTimeoutException`

GetCrawlerMetrics ação (Python: get_crawler_metrics)

Recupera métricas dos crawlers especificados.

Solicitação

- `CrawlerNameList` – Uma matriz de strings UTF-8, no máximo 100 strings.

  Uma lista dos nomes dos crawlers de onde as métricas serão recuperadas.

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo de uma lista a ser retornada.

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `CrawlerMetricsList` – Uma matriz de objetos CrawlerMetrics.

  Uma lista de métricas para o crawler especificado.

- `NextToken` – String UTF-8.

  Um token de continuação, se a lista retornada não contiver a métrica mais recente disponível.

Erros

- `OperationTimeoutException`

UpdateCrawler ação (Python: update_crawler)

Atualiza um crawler. Se um crawler estiver em execução, você precisará interrompê-lo usando `StopCrawler` antes de fazer a atualização.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do novo crawler.

- `Role` – String UTF-8.

  A função do IAM ou o nome de recurso da Amazon (ARN) de uma função do IAM que é usada pelo novo crawler para acessar os recursos do cliente.

- `DatabaseName` – String UTF-8.

  O AWS Glue banco de dados onde os resultados são armazenados, tal como:`arn:aws:daylight:us-east-1::database/sometable/*`.

- `Description` – String UTF-8 com comprimento não superior a 2048 bytes, correspondente a URI address multi-line string pattern.

  Uma descrição do novo crawler.

- `Targets` – Um objeto CrawlerTargets.

  Uma lista de destinos a serem rastreados.

- `Schedule` – String UTF-8.

  Uma expressão `cron` usada para especificar a programação (consulte Programações baseadas em hora para tarefas e crawlers. Por exemplo, para executar algo todos os dias às 12h15 UTC, especifique: `cron(15 12 * * ? *)`.

- `Classifiers` – Uma matriz de strings UTF-8.

Uma lista de classificadores personalizados que o usuário registrou. Por padrão, todos os classificadores integrados são incluídos em um rastreamento. No entanto, esses classificadores personalizados sempre substituem os classificadores padrão de uma determinada classificação.

- `TablePrefix` – String UTF-8 com comprimento não superior a 128 bytes.

  O prefixo que é usado para tabelas de catálogo criadas.

- `SchemaChangePolicy` – Um objeto SchemaChangePolicy.

  A política do comportamento de atualização e exclusão do crawler.

- `RecrawlPolicy` – Um objeto RecrawlPolicy.

  Uma política que especifica se deseja rastrear todo o conjunto de dados novamente ou rastrear somente pastas que foram adicionadas desde a última execução do crawler.

- `LineageConfiguration` – Um objeto LineageConfiguration.

  Especifica as definições de configuração de linhagem de dados para o crawler.

- `LakeFormationConfiguration` – Um objeto LakeFormationConfiguration.

  Especifica AWS Lake Formation as configurações do rastreador.

- `Configuration` – String UTF-8.

  Informações de configuração do crawler. Esta string JSON com versionamento permite que os usuários especifiquem os aspectos do comportamento de um crawler. Para obter mais informações, consulte Definir opções de configuração do crawler.

- `CrawlerSecurityConfiguration` – String UTF-8 com comprimento não superior a 128 bytes.

  O nome da estrutura `SecurityConfiguration` a ser usada por este crawler.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InvalidInputException`
- `VersionMismatchException`

- `EntityNotFoundException`
- `CrawlerRunningException`
- `OperationTimeoutException`

StartCrawler ação (Python: start_crawler)

Inicia um rastreamento usando o crawler especificado, independentemente do que estiver programado. Se o rastreador já estiver em execução, retornará a. CrawlerRunningException

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do crawler a ser iniciado.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `CrawlerRunningException`
- `OperationTimeoutException`

StopCrawler ação (Python: stop_crawler)

Se o crawler especificado estiver em execução, o rastreamento será interrompido.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do crawler a ser interrompido.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `CrawlerNotRunningException`
- `CrawlerStoppingException`
- `OperationTimeoutException`

BatchGetCrawlers ação (Python: batch_get_crawlers)

Retorna uma lista de metadados do recurso para uma lista de nomes de crawler. Depois de chamar a operação `ListCrawlers`, você pode chamar essa operação para acessar os dados aos quais você recebeu permissões. Essa operação oferece suporte a todas as permissões do IAM, incluindo condições de permissão que usam tags.

Solicitação

- `CrawlerNames` – Obrigatório: uma matriz de strings UTF-8, no máximo 100 strings.

  Uma lista de nomes de crawler, que podem ser os nomes retornados da operação `ListCrawlers`.

Resposta

- `Crawlers` – Uma matriz de objetos Crawler.

  Uma lista de definições do crawler.
- `CrawlersNotFound` – Uma matriz de strings UTF-8, no máximo 100 strings.

  Uma lista de nomes de crawlers que não foram encontrados.

Erros

- `InvalidInputException`

- `OperationTimeoutException`

ListCrawlers ação (Python: list_crawlers)

Recupera os nomes de todos os recursos do rastreador nessa AWS conta ou os recursos com a tag especificada. Essa operação permite que você veja quais recursos estão disponíveis em sua conta e seus nomes.

Essa operação aceita o campo `Tags` opcional, que pode ser usado como um filtro na resposta, para que recursos com tags possam ser recuperados como um grupo. Se você optar por usar a filtragem por tags, apenas os recursos com a tag serão recuperados.

Solicitação

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo de uma lista a ser retornada.
- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma solicitação de continuação.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  Especifica apenas o retorno desses recursos com tags.

Resposta

- `CrawlerNames` – Uma matriz de strings UTF-8, no máximo 100 strings.

  Os nomes de todos os crawlers na conta ou os crawlers com as tags especificadas.
- `NextToken` – String UTF-8.

  Um token de continuação, se a lista retornada não contiver a métrica mais recente disponível.

Erros

- `OperationTimeoutException`

ListCrawls ação (Python: list_crawls)

Retorna todos os crawls de um crawler especificado. Retorna apenas os crawls que ocorreram desde a data de execução do recurso de histórico do crawler e retém apenas até 12 meses de crawls. Crawls mais antigos não serão retornados.

Você pode usar essa API para:

- Recuperar todos os crawls de um crawler especificado.
- Recuperar todos os crawls de um crawler especificado dentro de uma contagem limitada.
- Recuperar todos os crawls de um crawler especificado em um intervalo de tempo específico.
- Recuperar todos os crawls de um crawler especificado com um determinado estado, ID de crawl ou valor de hora de DPU.

Solicitação

- `CrawlerName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do crawler cujas execuções você deseja recuperar.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de resultados a serem retornados. O padrão é 20 e o máximo é 100.
- `Filters` – Uma matriz de objetos CrawlsFilter.

  Filtra os crawls de acordo com os critérios especificados em uma lista de objetos de `CrawlsFilter`.
- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `Crawls` – Uma matriz de objetos CrawlerHistory.

Uma lista de objetos de `CrawlerHistory` que representam as execuções de crawl que satisfazem seus critérios.

- `NextToken` – String UTF-8.

  Um token de continuação para paginação da lista de tokens retornada, retornado se o segmento atual da lista não for o último.

Erros

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`

API de estatísticas de colunas

A API de estatísticas de colunas descreve as APIs do AWS Glue para retornar estatísticas em colunas em uma tabela.

Tipos de dados

- Estrutura ColumnStatisticsTaskRun
- Estrutura ColumnStatisticsTaskRunningException
- Estrutura ColumnStatisticsTaskNotRunningException
- Estrutura ColumnStatisticsTaskStoppingException

Estrutura ColumnStatisticsTaskRun

O objeto que mostra os detalhes da execução das estatísticas da coluna.

Campos

- `CustomerId` – String UTF-8, não mais de 12 bytes.

  O ID da conta da AWS.

- `ColumnStatisticsTaskRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O identificador para a execução da tarefa de estatísticas de coluna específica.

- `DatabaseName` – String UTF-8.

  O banco de dados em que a tabela reside.

- `TableName` – String UTF-8.

  O nome da tabela para a qual as estatísticas da coluna são geradas.

- `ColumnNameList` – Uma matriz de strings UTF-8.

  Uma lista de nomes de coluna. Se nenhum valor for fornecido, todos os nomes de colunas da tabela serão usados por padrão.

- `CatalogID` – String de ID de catálogo, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual reside a tabela. Se nenhum valor for fornecido, o ID da conta da AWS será usado por padrão.

- `Role` – String UTF-8.

  O perfil do IAM assumido pelo serviço para gerar estatísticas.

- `SampleSize`: número (double), no máximo 100.

  O percentual de linhas usadas para gerar estatísticas. Se nenhum valor for fornecido, a tabela inteira será usada para gerar estatísticas.

- `SecurityConfiguration` – String UTF-8 com comprimento não superior a 128 bytes.

  Nome da configuração de segurança usada para criptografar os logs do CloudWatch para a execução da tarefa de estatísticas da coluna.

- `NumberOfWorkers` – Número (inteiro), pelo menos 1.

  O número de operadores usados para gerar estatísticas de coluna. O trabalho é pré-configurado para escalar automaticamente para até 25 instâncias.

- `WorkerType` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O tipo dos operadores usados para gerar estatísticas. O padrão é `g.1x`.

- `Status` – String UTF-8 (valores válidos: `STARTING` | `RUNNING` | `SUCCEEDED` | `FAILED` | `STOPPED`).

O status da execução da tarefa.

- `CreationTime` – Timestamp.

  O horário em que a tarefa foi criada.

- `LastUpdated` – Timestamp.

  O momento em que a tarefa foi modificada pela última vez.

- `StartTime` – Timestamp.

  A hora de início da tarefa.

- `EndTime` – Timestamp.

  A hora de término da tarefa.

- `ErrorMessage` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A mensagem de erro para o trabalho.

- `DPUSeconds` – Número (dobro), não mais do que None (Nenhum).

  O uso calculado da DPU em segundos para todos os operadores com escalonamento automático.

Estrutura ColumnStatisticsTaskRunningException

Uma exceção lançada quando você tenta iniciar outro trabalho enquanto executa um trabalho de geração de estatísticas de coluna.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura ColumnStatisticsTaskNotRunningException

Uma exceção lançada quando você tenta interromper a execução de uma tarefa quando não há nenhuma tarefa em execução.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura ColumnStatisticsTaskStoppingException

Uma exceção lançada quando você tenta interromper a execução de uma tarefa.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Operações

- Ação StartColumnStatisticsTaskRun (Python: start_column_statistics_task_run)
- Ação GetColumnStatisticsTaskRun (Python: get_column_statistics_task_run)
- Ação GetColumnStatisticsTaskRuns (Python: get_column_statistics_task_runs)
- Ação ListColumnStatisticsTaskRuns (Python: list_column_statistics_task_runs)
- Ação StopColumnStatisticsTaskRun (Python: stop_column_statistics_task_run)

Ação StartColumnStatisticsTaskRun (Python: start_column_statistics_task_run)

Inicia a execução de uma tarefa de estatísticas de colunas para uma tabela e colunas especificadas.

Solicitação

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no qual a tabela reside.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela para gerar estatísticas.

- `ColumnNameList` – Uma matriz de strings UTF-8.

  Uma lista de nomes de colunas para gerar estatísticas. Se nenhum valor for fornecido, todos os nomes de colunas da tabela serão usados por padrão.

- `Role` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O perfil do IAM assumido pelo serviço para gerar estatísticas.

- `SampleSize`: número (double), no máximo 100.

  O percentual de linhas usadas para gerar estatísticas. Se nenhum valor for fornecido, a tabela inteira será usada para gerar estatísticas.

- `CatalogID` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo de dados no qual a tabela reside. Se nenhum valor for fornecido, o ID da conta da AWS será usado por padrão.

- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome da configuração de segurança usada para criptografar os logs do CloudWatch para a execução da tarefa de estatísticas da coluna.

Resposta

- `ColumnStatisticsTaskRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador para a execução da tarefa de estatísticas de colunas.

Erros

- `AccessDeniedException`
- `EntityNotFoundException`
- `ColumnStatisticsTaskRunningException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

- `InvalidInputException`

Ação GetColumnStatisticsTaskRun (Python: get_column_statistics_task_run)

Obtenha os metadados/informações associados para a execução de uma tarefa com um ID de execução de tarefa específico.

Solicitação

- `ColumnStatisticsTaskRunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador para a execução da tarefa de estatísticas de coluna específica.

Resposta

- `ColumnStatisticsTaskRun` – Um objeto ColumnStatisticsTaskRun.

  Um `ColumnStatisticsTaskRun` objeto que representa os detalhes da execução de estatísticas de colunas.

Erros

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`

Ação GetColumnStatisticsTaskRuns (Python: get_column_statistics_task_runs)

Recupera informações sobre todas as execuções associadas à tabela especificada.

Solicitação

- `DatabaseName` – Obrigatório: string UTF-8.

  O nome do banco de dados no qual a tabela reside.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome da tabela.

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo da resposta.

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `ColumnStatisticsTaskRuns` – Uma matriz de objetos ColumnStatisticsTaskRun.

  Uma lista de execuções de tarefas de estatísticas de colunas.

- `NextToken` – String UTF-8.

  Um token de continuação, se todas as execuções de tarefas ainda não tiverem sido retornadas.

Erros

- `OperationTimeoutException`

Ação ListColumnStatisticsTaskRuns (Python: list_column_statistics_task_runs)

Liste todas as execuções tarefas para uma conta específica.

Solicitação

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo da resposta.

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `ColumnStatisticsTaskRunIds` – Uma matriz de strings UTF-8, no máximo 100 strings.

Uma lista IDs de execuções de tarefas de estatísticas de colunas.

- `NextToken` – String UTF-8.

  Um token de continuação, se todos os IDs de execução de tarefas ainda não tiverem sido retornados.

Erros

- `OperationTimeoutException`

Ação StopColumnStatisticsTaskRun (Python: stop_column_statistics_task_run)

Interrompe a execução de uma tarefa para a tabela especificada.

Solicitação

- `DatabaseName` – Obrigatório: string UTF-8.

  O nome do banco de dados no qual a tabela reside.

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `ColumnStatisticsTaskNotRunningException`
- `ColumnStatisticsTaskStoppingException`
- `OperationTimeoutException`

API do programador do crawler

A API Crawler scheduler descreve os tipos de dados de crawler do AWS Glue, juntamente com a API para criar, excluir, atualizar e listar crawlers.

Tipos de dados

- Estrutura Schedule

Estrutura Schedule

Um objeto de programação que usa uma instrução `cron` para programar um evento.

Campos

- `ScheduleExpression` – String UTF-8.

  Uma expressão `cron` usada para especificar a programação (consulte Programações baseadas em hora para tarefas e crawlers. Por exemplo, para executar algo todos os dias às 12h15 UTC, especifique: `cron(15 12 * * ? *)`.
- `State` – String UTF-8 (valores válidos: `SCHEDULED` | `NOT_SCHEDULED` | `TRANSITIONING`).

  O estado da programação.

Operações

- Ação UpdateCrawlerSchedule (Python: update_crawler_schedule)
- Ação StartCrawlerSchedule (Python: start_crawler_schedule)
- Ação StopCrawlerSchedule (Python: stop_crawler_schedule)

Ação UpdateCrawlerSchedule (Python: update_crawler_schedule)

Atualiza a programação de um crawler usando uma expressão `cron`.

Solicitação

- `CrawlerName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome do crawler cuja programação será atualizada.

- `Schedule` – String UTF-8.

  A expressão `cron` atualizada usada para especificar a programação (consulte Programações baseadas em hora para tarefas e crawlers). Por exemplo, para executar algo todos os dias às 12h15 UTC, especifique: `cron(15 12 * * ? *)`.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `VersionMismatchException`
- `SchedulerTransitioningException`
- `OperationTimeoutException`

Ação StartCrawlerSchedule (Python: start_crawler_schedule)

Altera o estado da programação do crawler especificado para `SCHEDULED`, a menos que ele já esteja em execução ou o estado da programação já seja `SCHEDULED`.

Solicitação

- `CrawlerName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do crawler a ser programado.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `SchedulerRunningException`
- `SchedulerTransitioningException`
- `NoScheduleException`
- `OperationTimeoutException`

Ação StopCrawlerSchedule (Python: stop_crawler_schedule)

Define o estado da programação do crawler especificado para `NOT_SCHEDULED`, mas não o interrompe se já estiver em execução.

Solicitação

- `CrawlerName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do crawler cujo estado da programação será definido.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `SchedulerNotRunningException`
- `SchedulerTransitioningException`
- `OperationTimeoutException`

API de scripts de ETL de geração automática

A API de geração de script de ETL descreve os tipos de dados e a API para gerar scripts de ETL no AWS Glue.

Tipos de dados

Estrutura CodeGenNode

Representa um nó em um gráfico acíclico dirigido (DAG)

Campos

- `Id` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Identifier string pattern.

  Um identificador de nó que é exclusivo no gráfico do nó.
- `NodeType` – Obrigatório: string UTF-8.

  Qual é o tipo de nó.
- `Args` – Obrigatório: uma matriz de CodeGenNodeArg objetos, não mais de 50 estruturas.

  Propriedades do nó, na forma de pares nome-valor.
- `LineNumber` – Número (íntegro).

  O número de linha do nó.

Estrutura CodeGenNodeArg

Argumento ou propriedade de um nó.

Campos

- `Name` – Obrigatório: string UTF-8.

O nome do argumento ou da propriedade.

- `Value` – Obrigatório: string UTF-8.

  O valor do argumento ou da propriedade.

- `Param` – Booleano.

  True, se o valor for usado como um parâmetro.

Estrutura CodeGenEdge

Representa um ponto direcional em um gráfico acíclico dirigido (DAG).

Campos

- `Source` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Identifier string pattern.

  O ID do nó onde o ponto será iniciado.

- `Target` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Identifier string pattern.

  O ID do nó onde o ponto será encerrado.

- `TargetParameter` – String UTF-8.

  O destino do ponto.

Estrutura Location

O local dos recursos.

Campos

- `Jdbc` – Uma matriz de CodeGenNodeArg objetos, não mais de 50 estruturas.

  Um local JDBC.

- `S3` – Uma matriz de CodeGenNodeArg objetos, não mais de 50 estruturas.

  Uma localização do Amazon Simple Storage Service (Amazon S3).

- `DynamoDB` – Uma matriz de CodeGenNodeArg objetos, não mais de 50 estruturas.

  O local de uma tabela do Amazon DynamoDB.

Estrutura CatalogEntry

Especifica uma definição de tabela no AWS Glue Data Catalog.

Campos

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O banco de dados em que os metadados de tabela residem.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da tabela em questão.

Estrutura MappingEntry

Define um mapeamento.

Campos

- `SourceTable` – String UTF-8.

  O nome da tabela de origem.
- `SourcePath` – String UTF-8.

  O caminho de origem do .
- `SourceType` – String UTF-8.

  O tipo de origem.
- `TargetTable` – String UTF-8.

  O nome da tabela.
- `TargetPath` – String UTF-8.

O caminho de destino.

- `TargetType` – String UTF-8.

  O tipo de destino.

Operações

- Ação CreateScript (Python: create_script)
- Ação GetDataflowGraph (Python: get_dataflow_graph)
- Ação GetMapping (Python: get_mapping)
- Ação GetPlan (Python: get_plan)

Ação CreateScript (Python: create_script)

Transforma um gráfico acíclico dirigido (DAG) em código.

Solicitação

- `DagNodes` – Uma matriz de objetos CodeGenNode.

  Uma lista de nós no DAG.

- `DagEdges` – Uma matriz de objetos CodeGenEdge.

  Uma lista de pontos no DAG.

- `Language`: string UTF-8 (valores válidos: `PYTHON` | `SCALA`).

  A linguagem de programação do código resultante do DAG.

Resposta

- `PythonScript` – String UTF-8.

  O script Python gerado a partir do DAG.

- `ScalaCode` – String UTF-8.

  O código Scala gerado a partir do DAG.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetDataflowGraph (Python: get_dataflow_graph)

Transforma um script Python em um gráfico acíclico dirigido (DAG).

Solicitação

- `PythonScript` – String UTF-8.

  O script Python a ser transformado.

Resposta

- `DagNodes` – Uma matriz de objetos CodeGenNode.

  Uma lista de nós no DAG resultante.
- `DagEdges` – Uma matriz de objetos CodeGenEdge.

  Uma lista de pontos no DAG resultante.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetMapping (Python: get_mapping)

Cria mapeamentos.

Solicitação

- `Source` – Obrigatório: um objeto CatalogEntry.

Especifica a tabela de origem.

- `Sinks` – Uma matriz de objetos CatalogEntry.

  Uma lista de tabelas de destino.

- `Location` – Um objeto Local.

  Parâmetros para o mapeamento.

Resposta

- `Mapping` – Obrigatório: uma matriz de objetos MappingEntry.

  Uma lista de mapeamentos para os destinos especificados.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Ação GetPlan (Python: get_plan)

Obtém o código para executar um mapeamento especificado.

Solicitação

- `Mapping` – Obrigatório: uma matriz de objetos MappingEntry.

  A lista de mapeamentos de uma tabela de origem para tabelas de destino.

- `Source` – Obrigatório: um objeto CatalogEntry.

  A tabela de origem.

- `Sinks` – Uma matriz de objetos CatalogEntry.

  O nome das tabelas.

- `Location` – Um objeto Local.

  Os parâmetros para o mapeamento.

- `Language`: string UTF-8 (valores válidos: `PYTHON` | `SCALA`).

  A linguagem de programação do código para executar o mapeamento.

- `AdditionalPlanOptionsMap` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Um mapa para conter parâmetros de chave-valor opcionais extras.

  Atualmente, estes pares de chave-valor são suportados:

  - `inferSchema`: especifica se é para definir `inferSchema` como true ou false para o script padrão gerado por um trabalho do AWS Glue. Por exemplo, para definir `inferSchema` como true, passe o seguinte par de chave-valor:

    ```
    --additional-plan-options-map '{"inferSchema":"true"}'
    ```

Resposta

- `PythonScript` – String UTF-8.

  Um script Python para executar o mapeamento.

- `ScalaCode` – String UTF-8.

  O código Scala para executar o mapeamento.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

API de trabalhos visuais

A API Visual job permite criar trabalhos de integração de dados usando a AWS Glue API de um objeto JSON que representa uma configuração visual de um AWS Glue trabalho.

Uma lista de `CodeGenConfigurationNodes` é fornecida a uma API de criação ou atualização de tarefas para registrar um DAG no AWS Glue Studio para a tarefa criada e gerar o código associado.

Tipos de dados

- Estrutura S3 HudiDirectTarget
- Estrutura S3 DeltaCatalogTarget
- Estrutura S3 DeltaDirectTarget
- DirectSchemaChangePolicy estrutura
- ApplyMapping estrutura
- Estrutura Mapping
- SelectFields estrutura
- DropFields estrutura
- RenameField estrutura
- Estrutura Spigot
- Estrutura Join
- JoinColumn estrutura
- SplitFields estrutura
- SelectFromCollection estrutura
- FillMissingValues estrutura
- Estrutura Filter
- FilterExpression estrutura
- FilterValue estrutura
- CustomCode estrutura
- Estrutura SparkSQL
- SqlAlias estrutura
- DropNullFields estrutura
- NullCheckBoxList estrutura
- NullValueField estrutura
- Estrutura Datatype
- Estrutura Merge
- Estrutura Union
- Estrutura PIIDetection
- Estrutura Aggregate
- DropDuplicates estrutura

- GovernedCatalogTarget estrutura
- GovernedCatalogSource estrutura
- AggregateOperation estrutura
- GlueSchema estrutura
- GlueStudioSchemaColumn estrutura
- GlueStudioColumn estrutura
- DynamicTransform estrutura
- TransformConfigParameter estrutura
- EvaluateDataQuality estrutura
- Estrutura DQ ResultsPublishingOptions
- Estrutura DQ StopJobOnFailureOptions
- EvaluateDataQualityMultiFrame estrutura
- Estrutura da fórmula
- RecipeReference estrutura
- SnowflakeNodeData estrutura
- SnowflakeSource estrutura
- SnowflakeTarget estrutura
- ConnectorDataSource estrutura
- ConnectorDataTarget estrutura

CodeGenConfigurationNode estrutura

O `CodeGenConfigurationNode` enumera todos os tipos de nós válidos. Uma e apenas uma de suas variáveis membro podem ser preenchidas.

Campos

- `AthenaConnectorSource` – Um objeto AthenaConnectorSource.

  Especifica um conector para uma fonte de dados do Amazon Athena.
- `JDBCConnectorSource` – Um objeto JDBC ConnectorSource.

  Especifica um conector para uma fonte de dados JDBC.
- `SparkConnectorSource` – Um objeto SparkConnectorSource.

Especifica um conector para uma fonte de dados do Apache Spark.

- `CatalogSource` – Um objeto CatalogSource.

  Especifica um armazenamento de dados no Catálogo AWS Glue de Dados.

- `RedshiftSource` – Um objeto RedshiftSource.

  Especifica um datastore do Amazon Redshift.

- `S3CatalogSource` – Um objeto S3 CatalogSource.

  Especifica um armazenamento de dados do Amazon S3 no AWS Glue catálogo de dados.

- `S3CsvSource` – Um objeto S3 CsvSource.

  Especifica um datastore CSV (valores separados por comando) armazenado no Amazon S3.

- `S3JsonSource` – Um objeto S3 JsonSource.

  Especifica um datastore JSON armazenado no Amazon S3.

- `S3ParquetSource` – Um objeto S3 ParquetSource.

  Especifica um datastore do Apache Parquet armazenado no Amazon S3.

- `RelationalCatalogSource` – Um objeto RelationalCatalogSource.

  Especifica um armazenamento de dados de catálogo relacional no Catálogo de AWS Glue Dados.

- `DynamoDBCatalogSource` – Um objeto DynamoDB CatalogSource.

  Especifica um armazenamento de dados do Catálogo do DynamoDBC no Catálogo de Dados. AWS Glue

- `JDBCConnectorTarget` – Um objeto JDBC ConnectorTarget.

  Especifica um destino de dados que grava no Amazon S3 no armazenamento colunar do Apache Parquet.

- `SparkConnectorTarget` – Um objeto SparkConnectorTarget.

  Especifica um destino que usa um conector Apache Spark.

- `CatalogTarget` – Um objeto BasicCatalogTarget.

  Especifica um destino que usa uma tabela do Catálogo AWS Glue de Dados.

- `RedshiftTarget` – Um objeto RedshiftTarget.

Especifica um destino que usa o Amazon Redshift.

- `S3CatalogTarget` – Um objeto S3 CatalogTarget.

  Especifica um destino de dados que grava no Amazon S3 usando AWS Glue o catálogo de dados.

- `S3GlueParquetTarget` – Um objeto S3 GlueParquetTarget.

  Especifica um destino de dados que grava no Amazon S3 no armazenamento colunar do Apache Parquet.

- `S3DirectTarget` – Um objeto S3 DirectTarget.

  Especifica um destino de dados que grava no Amazon S3.

- `ApplyMapping` – Um objeto ApplyMapping.

  Especifica uma transformação que mapeia chaves de propriedade de dados na fonte dos dados para chaves de propriedade de dados no destino dos dados. Você pode renomear chaves, modificar os tipos de dados para chaves e escolher quais chaves remover do conjunto de dados.

- `SelectFields` – Um objeto SelectFields.

  Especifica uma transformação que escolhe as chaves de propriedade de dados que você deseja manter.

- `DropFields` – Um objeto DropFields.

  Especifica uma transformação que escolhe as chaves de propriedade de dados que você deseja descartar.

- `RenameField` – Um objeto RenameField.

  Especifica uma transformação que renomeia uma única chave de propriedade de dados.

- `Spigot` – Um objeto Spigot.

  Especifica uma transformação que grava amostras dos dados em um bucket do Amazon S3.

- `Join` – Um objeto Ingressar.

  Especifica uma transformação que une dois conjuntos de dados em um só, usando uma frase de comparação nas chaves de propriedade de dados especificadas. Você pode usar junção inner (interna), outer (externa), left (à esquerda), right (à direita), left semi (semi à esquerda) e left anti (anti à esquerda).

- `SplitFields` – Um objeto SplitFields.

Especifica uma transformação que divide chaves de propriedade de dados em dois `DynamicFrames`. A saída é uma coleção de `DynamicFrames`: um com chaves de propriedade de dados selecionadas e outro com as chaves de propriedade de dados restantes.

- `SelectFromCollection` – Um objeto SelectFromCollection.

  Especifica uma transformação que escolhe um `DynamicFrame` de uma coleção de `DynamicFrames`. A saída é o `DynamicFrame` selecionado.

- `FillMissingValues` – Um objeto FillMissingValues.

  Especifica uma transformação que localiza registros no conjunto de dados que tenham valores ausentes e adiciona um novo campo com um valor determinado por imputação. O conjunto de dados de entrada é usado para treinar o modelo de machine learning que determina qual deve ser o valor ausente.

- `Filter` – Um objeto Filtro.

  Especifica uma transformação que divide um conjunto de dados em dois, com base em uma condição de filtro.

- `CustomCode` – Um objeto CustomCode.

  Especifica uma transformação que usa código personalizado que você fornece para executar a transformação de dados. A saída é uma coleção de DynamicFrames.

- `SparkSQL` – Um objeto SparkSQL.

  Especifica uma transformação em que você insere uma consulta de SQL usando a sintaxe do Spark SQL para transformar os dados. A saída é um único `DynamicFrame`.

- `DirectKinesisSource` – Um objeto DirectKinesisSource.

  Especifica uma fonte de dados direta do Amazon Kinesis.

- `DirectKafkaSource` – Um objeto DirectKafkaSource.

  Especifica um datastore do Apache Kafka.

- `CatalogKinesisSource` – Um objeto CatalogKinesisSource.

  Especifica uma fonte de dados do Kinesis no AWS Glue catálogo de dados.

- `CatalogKafkaSource` – Um objeto CatalogKafkaSource.

  Especifica um datastore do Apache Kafka no catálogo de dados.

- `DropNullFields` – Um objeto DropNullFields.

  Especifica uma transformação que remove colunas do conjunto de dados se todos os valores na coluna forem 'null'. Por padrão, o AWS Glue Studio reconhecerá objetos nulos, mas alguns valores, como cadeias de caracteres vazias, sequências de caracteres “nulas”, números inteiros -1 ou outros espaços reservados, como zeros, não são automaticamente reconhecidos como nulos.
- `Merge` – Um objeto Mesclar.

  Especifica uma transformação que mescla um `DynamicFrame` com um `DynamicFrame` de preparação, de acordo com as chaves primárias especificadas para identificar registros. Registros duplicados (com as mesmas chaves primárias) não são eliminados.
- `Union` – Um objeto Union.

  Especifica uma transformação que combina as linhas de dois ou mais conjuntos de dados em um único resultado.
- `PIIDetection` – Um objeto PIIdetection.

  Especifica uma transformação que identifica, remove ou mascara dados de PII.
- `Aggregate` – Um objeto Agregar.

  Especifica uma transformação que agrupa linhas por campos escolhidos e calcula o valor agregado por função especificada.
- `DropDuplicates` – Um objeto DropDuplicates.

  Especifica uma transformação que remove linhas de dados repetidos de um conjunto de dados.
- `GovernedCatalogTarget` – Um objeto GovernedCatalogTarget.

  Especifica um destino de dados que grava em um catálogo governado.
- `GovernedCatalogSource` – Um objeto GovernedCatalogSource.

  Especifica uma fonte de dados em um catálogo de dados governado.
- `MicrosoftSQLServerCatalogSource` – Um objeto Microsoft SQL ServerCatalogSource.

  Especifica uma fonte de dados do Microsoft SQL Server no AWS Glue Data Catalog.
- `MySQLCatalogSource` – Um objeto MySQL CatalogSource.

  Especifica uma fonte de dados MySQL no AWS Glue Catálogo de Dados.
- `OracleSQLCatalogSource` – Um objeto Oracle SQL CatalogSource.

Especifica uma fonte de dados Oracle no Catálogo AWS Glue de Dados.

- `PostgreSQLCatalogSource` – Um objeto PostgreSQL CatalogSource.

  Especifica uma fonte de dados PostgresSQL no Catálogo de Dados. AWS Glue

- `MicrosoftSQLServerCatalogTarget` – Um objeto Microsoft SQL ServerCatalogTarget.

  Especifica um destino que usa o Microsoft SQL.

- `MySQLCatalogTarget` – Um objeto MySQL CatalogTarget.

  Especifica um destino que usa o MySQL.

- `OracleSQLCatalogTarget` – Um objeto Oracle SQL CatalogTarget.

  Especifica um destino que usa o Oracle SQL.

- `PostgreSQLCatalogTarget` – Um objeto PostgreSQL CatalogTarget.

  Especifica um destino que usa o Postgres SQL.

- `DynamicTransform` – Um objeto DynamicTransform.

  Especifica uma transformação visual personalizada criada por um usuário.

- `EvaluateDataQuality` – Um objeto EvaluateDataQuality.

  Especifica os critérios da avaliação de qualidade dos dados.

- `S3CatalogHudiSource` – Um objeto S3 CatalogHudiSource.

  Especifica uma fonte de dados Hudi que está registrada no Catálogo de AWS Glue Dados. A fonte de dados deve ser armazenada em Amazon S3.

- `CatalogHudiSource` – Um objeto CatalogHudiSource.

  Especifica uma fonte de dados Hudi que está registrada no Catálogo de AWS Glue Dados.

- `S3HudiSource` – Um objeto S3 HudiSource.

  Especifica uma fonte de dados Hudi armazenada em. Amazon S3

- `S3HudiCatalogTarget` – Um objeto S3 HudiCatalogTarget.

  Especifica um destino que grava em uma fonte de dados Hudi no Catálogo de AWS Glue Dados.

- `S3HudiDirectTarget` – Um objeto S3 HudiDirectTarget.

Especifica um destino que grava em uma fonte de dados Hudi em. Amazon S3

- `S3CatalogDeltaSource` – Um objeto S3 CatalogDeltaSource.

  Especifica uma fonte de dados do Delta Lake que está registrada no Catálogo AWS Glue de Dados. A fonte de dados deve ser armazenada em Amazon S3.

- `CatalogDeltaSource` – Um objeto CatalogDeltaSource.

  Especifica uma fonte de dados do Delta Lake que está registrada no Catálogo AWS Glue de Dados.

- `S3DeltaSource` – Um objeto S3 DeltaSource.

  Especifica uma fonte de dados do Delta Lake armazenada em Amazon S3.

- `S3DeltaCatalogTarget` – Um objeto S3 DeltaCatalogTarget.

  Especifica um destino que grava em uma fonte de dados do Delta Lake no Catálogo AWS Glue de Dados.

- `S3DeltaDirectTarget` – Um objeto S3 DeltaDirectTarget.

  Especifica um destino que grava em uma fonte de dados do Delta Lake em Amazon S3.

- `AmazonRedshiftSource` – Um objeto AmazonRedshiftSource.

  Especifica um destino que grava em uma fonte de dados no Amazon Redshift.

- `AmazonRedshiftTarget` – Um objeto AmazonRedshiftTarget.

  Especifica um destino que grava em um destino de dados no Amazon Redshift.

- `EvaluateDataQualityMultiFrame` – Um objeto EvaluateDataQualityMultiFrame.

  Especifica os critérios da avaliação de qualidade dos dados. Permite vários dados de entrada e retorna um conjunto de quadros dinâmicos.

- `Recipe` – Um objeto Fórmula.

  Especifica um nó de AWS Glue DataBrew receita.

- `SnowflakeSource` – Um objeto SnowflakeSource.

  Especifica uma fonte de dados do Snowflake.

- `SnowflakeTarget` – Um objeto SnowflakeTarget.

Especifica um destino que grava em uma fonte de dados do Snowflake.

- `ConnectorDataSource` – Um objeto ConnectorDataSource.

  Especifica uma fonte gerada com opções de conexão padrão.

- `ConnectorDataTarget` – Um objeto ConnectorDataTarget.

  Especifica um destino gerado com opções de conexão padrão.

Estrutura do JDBC ConnectorOptions

Opções de conexão adicionais para o conector.

Campos

- `FilterPredicate` – String UTF-8 correspondente a Custom string pattern #40.

  Cláusula de condição extra para filtrar dados da fonte. Por exemplo:

  ```
  BillingCity='Mountain View'
  ```

  Ao usar uma consulta em vez de um nome de tabela, você deve validar que a consulta funciona com o `filterPredicate` especificado.

- `PartitionColumn` – String UTF-8 correspondente a Custom string pattern #40.

  O nome de uma coluna de inteiros usada para o particionamento. Essa opção só funciona quando está incluída em `lowerBound`, `upperBound` e `numPartitions`. Essa opção funciona da mesma maneira que no leitor JDBC Spark SQL.

- `LowerBound`: número (inteiro longo), no máximo None (Nenhum).

  O valor mínimo de `partitionColumn` que é usado para decidir o passo de partição.

- `UpperBound`: número (inteiro longo), no máximo None (Nenhum).

  O valor máximo de `partitionColumn` que é usado para decidir o passo de partição.

- `NumPartitions`: número (inteiro longo), no máximo None (Nenhum).

  O número de partições. Esse valor, juntamente com `lowerBound` (inclusive) e `upperBound` (exclusive), forma os passos de partição para as expressões de cláusula `WHERE` geradas que são usadas para dividir a `partitionColumn`.

- `JobBookmarkKeys` – Uma matriz de strings UTF-8.

  O nome das chaves de marcador de trabalho pelas quais classificar.

- `JobBookmarkKeysSortOrder` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica ordem de classificação ascendente ou descendente.

- `DataTypeMapping` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string 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 é uma string UTF-8 (valores válidos: `DATE` | `STRING` | `TIMESTAMP` | `INT` | `FLOAT` | `LONG` | `BIGDECIMAL` | `BYTE` | `SHORT` | `DOUBLE`).

  Mapeamento de tipo de dados personalizado que constrói um mapeamento a partir de um tipo de dados JDBC para um tipo de dados do AWS Glue . Por exemplo, a opção `"dataTypeMapping":{"FLOAT":"STRING"}` mapeia campos de dados do tipo JDBC `FLOAT` para o `String` tipo Java chamando o `ResultSet.getString()` método do driver e o usa para criar o AWS Glue registro. O objeto `ResultSet` é implantado por cada driver, portanto, o comportamento é específico para o driver que você usa. Consulte a documentação do driver do JDBC para entender como ele executa as conversões.

StreamingDataPreviewOptions estrutura

Especifica opções relacionadas à previsualização de dados para exibir uma amostra de seus dados.

Campos

- `PollingTime`: número (longo), pelo menos 10.

  O tempo de sondagem, em milissegundos.

- `RecordPollingLimit`: número (longo), pelo menos 1.

  O limite para o número de registros sondados.

AthenaConnectorSource estrutura

Especifica um conector para uma fonte de dados do Amazon Athena.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.
- `ConnectionName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da conexão associada ao conector.
- `ConnectorName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome de um conector que ajuda a acessar o armazenamento de dados no AWS Glue Studio.
- `ConnectionType` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O tipo de conexão, como marketplace.athena ou custom.athena, designando uma conexão com um datastore do Amazon Athena.
- `ConnectionTable` – String UTF-8 correspondente a Custom string pattern #41.

  O nome da tabela na fonte de dados.
- `SchemaName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do grupo de logs do CloudWatch a ser lido. Por exemplo, `/aws-glue/jobs/output`.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte do Athena personalizada.

Estrutura do JDBC ConnectorSource

Especifica um conector para uma fonte de dados JDBC.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.
- `ConnectionName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

O nome da conexão associada ao conector.

- `ConnectorName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome de um conector que ajuda a acessar o armazenamento de dados no AWS Glue Studio.

- `ConnectionType` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O tipo de conexão, como marketplace.jdbc ou custom.jdbc, designando uma conexão com um datastore JDBC.

- `AdditionalOptions` – Um objeto JDBC ConnectorOptions.

  Opções de conexão adicionais para o conector.

- `ConnectionTable` – String UTF-8 correspondente a Custom string pattern #41.

  O nome da tabela na fonte de dados.

- `Query` – String UTF-8 correspondente a Custom string pattern #42.

  A tabela ou consulta SQL da qual obter os dados. Você pode especificar `ConnectionTable` ou `query`, mas não os dois.

- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte do JDBC personalizada.

SparkConnectorSource estrutura

Especifica um conector para uma fonte de dados do Apache Spark.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.

- `ConnectionName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da conexão associada ao conector.

- `ConnectorName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome de um conector que ajuda a acessar o armazenamento de dados no AWS Glue Studio.

- `ConnectionType` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O tipo de conexão, como marketplace.spark ou custom.spark, designando uma conexão com um datastore do Apache Spark.
- `AdditionalOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Opções de conexão adicionais para o conector.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte do Spark personalizada.

CatalogSource estrutura

Especifica um armazenamento de dados no Catálogo AWS Glue de Dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do datastore.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.

Estrutura do MySQL CatalogSource

Especifica uma fonte de dados MySQL no AWS Glue Catálogo de Dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.

Estrutura do PostgreSQL CatalogSource

Especifica uma fonte de dados PostgresSQL no Catálogo de Dados. AWS Glue

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.

Estrutura do OracleSQL CatalogSource

Especifica uma fonte de dados Oracle no Catálogo AWS Glue de Dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.

Estrutura Microsoft SQL ServerCatalogSource

Especifica uma fonte de dados do Microsoft SQL Server no AWS Glue Data Catalog.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.

CatalogKinesisSource estrutura

Especifica uma fonte de dados do Kinesis no AWS Glue catálogo de dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.
- `WindowSize` – Número (inteiro), não mais do que None (Nenhum).

  A quantidade de tempo gasto no processamento de cada micro lote.
- `DetectSchema` – Booleano.

  Se o esquema deve ser determinado automaticamente a partir dos dados recebidos.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.
- `StreamingOptions` – Um objeto KinesisStreamingSourceOptions.

Opções adicionais para a fonte de dados de transmissão do Kinesis.

- `DataPreviewOptions` – Um objeto StreamingDataPreviewOptions.

  Opções adicionais para previsualização de dados.

DirectKinesisSource estrutura

Especifica uma fonte de dados direta do Amazon Kinesis.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.

- `WindowSize` – Número (inteiro), não mais do que None (Nenhum).

  A quantidade de tempo gasto no processamento de cada micro lote.

- `DetectSchema` – Booleano.

  Se o esquema deve ser determinado automaticamente a partir dos dados recebidos.

- `StreamingOptions` – Um objeto KinesisStreamingSourceOptions.

  Opções adicionais para a fonte de dados de transmissão do Kinesis.

- `DataPreviewOptions` – Um objeto StreamingDataPreviewOptions.

  Opções adicionais para previsualização de dados.

KinesisStreamingSourceOptions estrutura

Opções adicionais para a fonte de dados de transmissão do Amazon Kinesis.

Campos

- `EndpointUrl` – String UTF-8 correspondente a Custom string pattern #40.

  O URL do endpoint do Kinesis.

- `StreamName` – String UTF-8 correspondente a Custom string pattern #40.

O nome do fluxo de dados do Kinesis.

- `Classification` – String UTF-8 correspondente a Custom string pattern #40.

  Uma classificação opcional.

- `Delimiter` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica o caractere delimitador.

- `StartingPosition` – String UTF-8 (valores válidos: `latest="LATEST"` | `trim_horizon="TRIM_HORIZON"` | `earliest="EARLIEST"` | `timestamp="TIMESTAMP"`).

  A posição inicial no fluxo de dados do Kinesis de onde ler os dados. Os valores possíveis são `"latest"`, `"trim_horizon"`, `"earliest"` ou uma string de timestamp no formato UTC no padrão `yyyy-mm-ddTHH:MM:SSZ` (onde `Z` representa um desvio do fuso horário UTC com +/-). Por exemplo: "2023-04-04T08:00:00-04:00"). O valor padrão é `"latest"`.

  Observação: o uso de um valor que seja uma string de carimbo de data/hora no formato UTC para “StartingPosition” é suportado somente para AWS Glue a versão 4.0 ou posterior.

- `MaxFetchTimeInMs`: número (inteiro longo), no máximo None (Nenhum).

  O tempo máximo para o executor do trabalho ler registros referentes ao lote atual do fluxo de dados do Kinesis especificado em milissegundos (ms). Várias chamadas de API `GetRecords` podem ser feitas nesse período. O valor padrão é `1000`.

- `MaxFetchRecordsPerShard`: número (inteiro longo), no máximo None (Nenhum).

  O número máximo de registros a serem obtidos por fragmento no fluxo de dados do Kinesis por microlote. Observação: o cliente poderá exceder esse limite se o trabalho de streaming já tiver lido registros extras do Kinesis (na mesma chamada get-records). Se `MaxFetchRecordsPerShard` precisa ser rigoroso, então precisa ser um múltiplo de `MaxRecordPerRead`. O valor padrão é `100000`.

- `MaxRecordPerRead`: número (inteiro longo), no máximo None (Nenhum).

  O número máximo de registros a serem obtidos por fragmento no fluxo de dados do Kinesis em cada operação getRecords. O valor padrão é `10000`.

- `AddIdleTimeBetweenReads` – Booleano.

  Adiciona um atraso de tempo entre duas operações getRecords consecutivas. O valor padrão é `"False"`. Essa opção só pode ser configurada para o Glue versão 2.0 e posterior.

- `IdleTimeBetweenReadsInMs`: número (inteiro longo), no máximo None (Nenhum).

  O atraso mínimo entre duas operações getRecords consecutivas, especificado em ms. O valor padrão é `1000`. Essa opção só pode ser configurada para o Glue versão 2.0 e posterior.
- `DescribeShardInterval`: número (inteiro longo), no máximo None (Nenhum).

  O intervalo mínimo de tempo entre duas chamadas de ListShards API para que seu script considere a refragmentação. O valor padrão é `1s`.
- `NumRetries` – Número (inteiro), não mais do que None (Nenhum).

  O número máximo de novas tentativas para solicitações de API do Kinesis Data Streams. O valor padrão é `3`.
- `RetryIntervalMs`: número (inteiro longo), no máximo None (Nenhum).

  O período de espera (especificado em ms) antes de repetir a chamada da API Kinesis Data Streams. O valor padrão é `1000`.
- `MaxRetryIntervalMs`: número (inteiro longo), no máximo None (Nenhum).

  O período de espera máximo (especificado em ms) entre duas tentativas de uma chamada de API Kinesis Data Streams. O valor padrão é `10000`.
- `AvoidEmptyBatches` – Booleano.

  Evita a criação de um trabalho de microlote vazio verificando se há dados não lidos no fluxo de dados do Kinesis antes do lote ser iniciado. O valor padrão é `"False"`.
- `StreamArn` – String UTF-8 correspondente a Custom string pattern #40.

  O nome de recurso da Amazon (ARN) do fluxo de dados do Kinesis.
- `RoleArn` – String UTF-8 correspondente a Custom string pattern #40.

  O nome do recurso da Amazon (ARN) da função a ser assumida pelo uso do AWS Security Token Service (AWS STS). Essa função deve ter permissões para descrever ou ler operações de registro para o fluxo de dados do Kinesis. Você deve usar esse parâmetro ao acessar um fluxo de dados em uma conta diferente. Usado em conjunto com `"awsSTSSessionName"`.
- `RoleSessionName` – String UTF-8 correspondente a Custom string pattern #40.

  Um identificador para a sessão que assume a função usando o AWS STS. Você deve usar esse parâmetro ao acessar um fluxo de dados em uma conta diferente. Usado em conjunto com `"awsSTSRoleARN"`.

- `AddRecordTimestamp` – String UTF-8 correspondente a Custom string pattern #40.

  Quando essa opção for definida como "true", a saída de dados conterá uma coluna adicional denominada "__src_timestamp" que indica a hora que o registro correspondente é recebido pelo fluxo. O valor padrão é "false". Essa opção é compatível com a AWS Glue versão 4.0 ou posterior.

- `EmitConsumerLagMetrics` – String UTF-8 correspondente a Custom string pattern #40.

  Quando essa opção é definida como “verdadeira”, para cada lote, ela emitirá as métricas da duração entre o registro mais antigo recebido pelo stream e o horário em AWS Glue que ele chega. CloudWatch O nome da métrica é “glue.driver.streaming”. maxConsumerLagInMs”. O valor padrão é "false". Essa opção é compatível com o AWS Glue versão 4.0 ou posterior.

- `StartingTimestamp` – String UTF-8.

  O timestamp do registro no fluxo de dados do Kinesis para começar a ler os dados. Os valores possíveis são uma string de timestamp no formato UTC no padrão `yyyy-mm-ddTHH:MM:SSZ` (onde Z representa um desvio do fuso horário UTC com +/-). Por exemplo: "2023-04-04T08:00:00+08:00").

CatalogKafkaSource estrutura

Especifica um datastore do Apache Kafka no catálogo de dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do datastore.

- `WindowSize` – Número (inteiro), não mais do que None (Nenhum).

  A quantidade de tempo gasto no processamento de cada micro lote.

- `DetectSchema` – Booleano.

  Se o esquema deve ser determinado automaticamente a partir dos dados recebidos.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

O nome do banco de dados do qual a leitura será feita.

- `StreamingOptions` – Um objeto KafkaStreamingSourceOptions.

  Especifica as opções de transmissão.

- `DataPreviewOptions` – Um objeto StreamingDataPreviewOptions.

  Especifica opções relacionadas à previsualização de dados para exibir uma amostra de seus dados.

DirectKafkaSource estrutura

Especifica um datastore do Apache Kafka.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do datastore.

- `StreamingOptions` – Um objeto KafkaStreamingSourceOptions.

  Especifica as opções de transmissão.

- `WindowSize` – Número (inteiro), não mais do que None (Nenhum).

  A quantidade de tempo gasto no processamento de cada micro lote.

- `DetectSchema` – Booleano.

  Se o esquema deve ser determinado automaticamente a partir dos dados recebidos.

- `DataPreviewOptions` – Um objeto StreamingDataPreviewOptions.

  Especifica opções relacionadas à previsualização de dados para exibir uma amostra de seus dados.

KafkaStreamingSourceOptions estrutura

Opções adicionais para transmissões.

Campos

- `BootstrapServers` – String UTF-8 correspondente a Custom string pattern #40.

  Uma lista de URLs do servidor de bootstrap, por exemplo, como `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`. Essa opção deve ser especificada na chamada de API ou definida nos metadados da tabela no Data Catalog.

- `SecurityProtocol` – String UTF-8 correspondente a Custom string pattern #40.

  O protocolo usado para se comunicar com os agentes. Os valores possíveis são `"SSL"` ou `"PLAINTEXT"`.

- `ConnectionName` – String UTF-8 correspondente a Custom string pattern #40.

  O nome da conexão.

- `TopicName` – String UTF-8 correspondente a Custom string pattern #40.

  O nome do tópico conforme especificado no Apache Kafka. É necessário especificar pelo menos um `"topicName"`, `"assign"` ou `"subscribePattern"`.

- `Assign` – String UTF-8 correspondente a Custom string pattern #40.

  As `TopicPartitions` específicas a consumir. É necessário especificar pelo menos um `"topicName"`, `"assign"` ou `"subscribePattern"`.

- `SubscribePattern` – String UTF-8 correspondente a Custom string pattern #40.

  Uma string regex Java que identifica a lista de tópicos para assinar. É necessário especificar pelo menos um `"topicName"`, `"assign"` ou `"subscribePattern"`.

- `Classification` – String UTF-8 correspondente a Custom string pattern #40.

  Uma classificação opcional.

- `Delimiter` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica o caractere delimitador.

- `StartingOffsets` – String UTF-8 correspondente a Custom string pattern #40.

  A posição inicial no tópico do Kafka de onde ler os dados. Os valores possíveis são `"earliest"` ou `"latest"`. O valor padrão é `"latest"`.

- `EndingOffsets` – String UTF-8 correspondente a Custom string pattern #40.

O ponto final quando uma consulta em lote é encerrada. Os valores possíveis são `"latest"` ou uma string JSON que especifica um deslocamento final para cada `TopicPartition`.

- `PollTimeoutMs`: número (inteiro longo), no máximo None (Nenhum).

  O tempo limite em milissegundos para sondar dados do Kafka em executores de trabalho do Spark. O valor padrão é `512`.

- `NumRetries` – Número (inteiro), não mais do que None (Nenhum).

  O número de novas tentativas antes de falhar em obter os deslocamentos do Kafka. O valor padrão é `3`.

- `RetryIntervalMs`: número (inteiro longo), no máximo None (Nenhum).

  O tempo em milissegundos a se esperar antes de tentar novamente buscar os deslocamentos do Kafka. O valor padrão é `10`.

- `MaxOffsetsPerTrigger`: número (inteiro longo), no máximo None (Nenhum).

  O limite de taxa no número máximo de deslocamentos que são processados por intervalo do acionador. O número total especificado de deslocamentos é dividido proporcionalmente entre `topicPartitions` de diferentes volumes. O valor padrão é nulo, o que significa que o consumidor lê todos os deslocamentos até o deslocamento mais recente conhecido.

- `MinPartitions` – Número (inteiro), não mais do que None (Nenhum).

  O número mínimo desejado de partições a serem lidas do Kafka. O valor padrão é nulo, o que significa que o número de partições do Spark é igual ao número de partições do Kafka.

- `IncludeHeaders` – Booleano.

  Se cabeçalhos do Kafka devem ser incluídos. Quando a opção estiver definida como "true", a saída de dados conterá uma coluna adicional chamada "glue_streaming_kafka_headers" com o tipo `Array[Struct(key: String, value: String)]`. O valor padrão é “false”. Essa opção está disponível somente na AWS Glue versão 3.0 ou posterior.

- `AddRecordTimestamp` – String UTF-8 correspondente a Custom string pattern #40.

  Quando essa opção for definida como “true“, a saída de dados conterá uma coluna adicional denominada "__src_timestamp" que indica a hora que o registro correspondente é recebido pelo tópico. O valor padrão é "false". Essa opção é compatível com a AWS Glue versão 4.0 ou posterior.

- `EmitConsumerLagMetrics` – String UTF-8 correspondente a Custom string pattern #40.

  Quando essa opção é definida como “verdadeira”, para cada lote, ela emitirá as métricas da duração entre o registro mais antigo recebido pelo tópico e a hora em AWS Glue que ele chega. CloudWatch O nome da métrica é “glue.driver.streaming”. maxConsumerLagInMs”. O valor padrão é "false". Essa opção é compatível com o AWS Glue versão 4.0 ou posterior.

- `StartingTimestamp` – String UTF-8.

  O timestamp do registro no tópico do Kafka para começar a ler os dados. Os valores possíveis são uma string de timestamp no formato UTC no padrão `yyyy-mm-ddTHH:MM:SSZ` (onde Z representa um desvio do fuso horário UTC com +/-). Por exemplo: "2023-04-04T08:00:00+08:00").

  Somente um de `StartingTimestamp` ou `StartingOffsets` deve ser definido.

RedshiftSource estrutura

Especifica um datastore do Amazon Redshift.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do datastore do Amazon Redshift.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O banco de dados do qual a leitura será feita.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  A tabela do banco de dados da qual a leitura será feita.

- `RedshiftTmpDir` – String UTF-8 correspondente a Custom string pattern #40.

  O caminho do Amazon S3 onde dados temporários podem ser preparados ao serem copiados do banco de dados.

- `TmpDirIAMRole` – String UTF-8 correspondente a Custom string pattern #40.

  A função do IAM com permissões.

AmazonRedshiftSource estrutura

Especifica uma fonte do Amazon Redshift.

Campos

- `Name` – String UTF-8 correspondente a Custom string pattern #43.

  O nome da fonte do Amazon Redshift.
- `Data` – Um objeto AmazonRedshiftNodeData.

  Especifica os dados do nó da fonte do Amazon Reshift.

AmazonRedshiftNodeData estrutura

Especifica um nó do Amazon Redshift.

Campos

- `AccessType` – String UTF-8 correspondente a Custom string pattern #39.

  O tipo de acesso para a conexão do Redshift. Pode ser uma conexão direta ou conexões de catálogo.
- `SourceType` – String UTF-8 correspondente a Custom string pattern #39.

  O tipo de origem para especificar se uma determinada tabela é a fonte ou uma consulta personalizada.
- `Connection` – Um objeto Opção.

  A AWS Glue conexão com o cluster do Redshift.
- `Schema` – Um objeto Opção.

  O nome do esquema do Redshift ao trabalhar com uma conexão direta.
- `Table` – Um objeto Opção.

  O nome da tabela do Redshift ao trabalhar com uma conexão direta.
- `CatalogDatabase` – Um objeto Opção.

O nome do banco de AWS Glue dados do Catálogo de Dados ao trabalhar com um catálogo de dados.

- `CatalogTable` – Um objeto Opção.

  O nome da tabela do Catálogo de AWS Glue Dados ao trabalhar com um catálogo de dados.

- `CatalogRedshiftSchema` – String UTF-8.

  O nome do esquema do Redshift ao trabalhar com um catálogo de dados.

- `CatalogRedshiftTable` – String UTF-8.

  A tabela do banco de dados da qual a leitura será feita.

- `TempDir` – String UTF-8 correspondente a Custom string pattern #40.

  O caminho do Amazon S3 onde dados temporários podem ser preparados ao serem copiados do banco de dados.

- `IamRole` – Um objeto Opção.

  Opcional. O nome do perfil usado ao conectar com o S3. O perfil do IAM, quando deixado em branco, assumirá como padrão o perfil no trabalho.

- `AdvancedOptions` – Uma matriz de objetos AmazonRedshiftAdvancedOption.

  Valores opcionais ao se conectar ao cluster do Redshift.

- `SampleQuery` – String UTF-8.

  O SQL usado para buscar os dados de uma fonte do Redshift quando SourceType é “consulta”.

- `PreAction` – String UTF-8.

  O SQL usado antes de um MERGE ou APPEND com upsert ser executado.

- `PostAction` – String UTF-8.

  O SQL usado antes de um MERGE ou APPEND com upsert ser executado.

- `Action` – String UTF-8.

  Especifica como a gravação em um cluster do Redshift ocorrerá.

- `TablePrefix` – String UTF-8 correspondente a Custom string pattern #39.

  Especifica o prefixo de uma tabela.

- `Upsert` – Booleano.

  A ação usada no Redshift vai para o coletor ao fazer um APPEND.

- `MergeAction` – String UTF-8 correspondente a Custom string pattern #39.

  A ação usada para determinar como um MERGE em um coletor do Redshift será tratado.

- `MergeWhenMatched` – String UTF-8 correspondente a Custom string pattern #39.

  A ação usada para determinar como um MERGE em um coletor do Redshift será tratado quando um registro existente corresponder a um novo registro.

- `MergeWhenNotMatched` – String UTF-8 correspondente a Custom string pattern #39.

  A ação usada para determinar como um MERGE em um coletor do Redshift será tratado quando um registro existente não corresponder a um novo registro.

- `MergeClause` – String UTF-8.

  O SQL usado em uma mesclagem personalizada para lidar com registros correspondentes.

- `CrawlerConnection` – String UTF-8.

  Especifica o nome da conexão associada à tabela do catálogo usada.

- `TableSchema` – Uma matriz de objetos Opção.

  A matriz de saída do esquema para um determinado nó.

- `StagingTable` – String UTF-8.

  O nome da tabela de preparação temporária usada ao fazer um MERGE ou APPEND com upsert.

- `SelectedColumns` – Uma matriz de objetos Opção.

  A lista de nomes de colunas usada para determinar um registro correspondente ao fazer MERGE ou APPEND com upsert.

AmazonRedshiftAdvancedOption estrutura

Especifica um valor opcional ao se conectar ao cluster do Redshift.

Campos

- `Key` – String UTF-8.

A chave para a opção de conexão adicional.

- `Value` – String UTF-8.

  O valor para a opção de conexão adicional.

Estrutura Option

Especifica um valor de opção.

Campos

- `Value` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica o valor da opção.

- `Label` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica o rótulo da opção.

- `Description` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica a descrição dae opção.

Estrutura S3 CatalogSource

Especifica um armazenamento de dados do Amazon S3 no AWS Glue catálogo de dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do datastore.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O banco de dados do qual a leitura será feita.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  A tabela do banco de dados da qual a leitura será feita.

- `PartitionPredicate` – String UTF-8 correspondente a Custom string pattern #40.

As partições que satisfazem a esse predicado são excluídas. Os arquivos dentro do período de retenção nessas partições não são excluídos. Definido como `""`, vazio por padrão.

- `AdditionalOptions` – Um objeto S3 SourceAdditionalOptions.

  Especifica opções de conexão adicionais.

Estrutura S3 SourceAdditionalOptions

Especifica opções de conexão adicionais para o datastore do Amazon S3.

Campos

- `BoundedSize` – Número (extenso).

  Define o limite superior para o tamanho de destino do conjunto de dados em bytes que serão processados.

- `BoundedFiles` – Número (extenso).

  Define o limite superior para o número alvo de arquivos que serão processados.

Estrutura S3 CsvSource

Especifica um datastore CSV (valores separados por comando) armazenado no Amazon S3.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do datastore.

- `Paths`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de caminhos do Amazon S3 dos quais fazer a leitura.

- `CompressionType`: string UTF-8 (valores válidos: `gzip="GZIP"` | `bzip2="BZIP2"`).

  Especifica como os dados são compactados. Isso geralmente não é necessário se os dados tem uma extensão de arquivo padrão. Os possíveis valores são `"gzip"` e `"bzip"`).

- `Exclusions` – Uma matriz de strings UTF-8.

Uma string contendo uma lista JSON de padrões glob a excluir estilo Unix. Por exemplo, "[\"**.pdf \"]" exclui todos os arquivos PDF.

- `GroupSize` – String UTF-8 correspondente a Custom string pattern #40.

  O tamanho do grupo de destino em bytes. O padrão é calculado com base no tamanho de dados de entrada e o tamanho de seu cluster. Quando há menos de 50.000 arquivos de entrada, `"groupFiles"` deve ser definido como `"inPartition"` para poder entrar em vigor.

- `GroupFiles` – String UTF-8 correspondente a Custom string pattern #40.

  O agrupamento de arquivos é ativado por padrão quando a entrada contiver mais de 50.000 arquivos. Para habilitar o agrupamento com menos de 50.000 arquivos, defina esse parâmetro como "inPartition". Para desabilitar o agrupamento quando houver mais de 50.000 arquivos, defina esse parâmetro como `"none"`.

- `Recurse` – Booleano.

  Se definido como verdadeiro, recursivamente lê arquivos em todos os subdiretórios de acordo com os caminhos especificados.

- `MaxBand` – Número (inteiro), não mais do que None (Nenhum).

  Esta opção controla a duração, em milissegundos, após a qual a listagem do s3 provavelmente será consistente. Arquivos com registros de data e hora de modificação dentro dos últimos milissegundos do MaxBand são rastreados especialmente quando usados JobBookmarks para contabilizar a consistência eventual do Amazon S3. A maioria dos usuários não precisa definir essa opção. O valor padrão é 900.000 milissegundos, ou 15 minutos.

- `MaxFilesInBand` – Número (inteiro), não mais do que None (Nenhum).

  Esta opção especifica o número máximo de arquivos para salvar nos últimos maxBand segundos. Se esse número for excedido, os arquivos extras são ignorados e apenas processados na próxima execução do trabalho.

- `AdditionalOptions` – Um objeto S3 DirectSourceAdditionalOptions.

  Especifica opções de conexão adicionais.

- `Separator` – obrigatório: string UTF-8 (valores válidos: `comma="COMMA"` | `ctrla="CTRLA"` | `pipe="PIPE"` | `semicolon="SEMICOLON"` | `tab="TAB"`).

  Especifica o caractere delimitador. O padrão é uma vírgula: ",", mas qualquer outro caractere pode ser especificado.

- `Escaper` – String UTF-8 correspondente a Custom string pattern #41.

  Especifica um caractere a ser usado para escape. Essa opção é usada somente ao ler arquivos CSV. O valor padrão é none. Se ativado, o caractere que imediatamente segue é usado no estado em que se encontram, exceto para um pequeno conjunto de escapes conhecidos (`\n`, `\r`, `\t` e `\0`).

- `QuoteChar`: obrigatório: string UTF-8 (valores válidos: `quote="QUOTE"` | `quillemet="QUILLEMET"` | `single_quote="SINGLE_QUOTE"` | `disabled="DISABLED"`).

  Especifica o caractere a ser usado para aspas. O padrão é aspas duplas: `'"'`. Defina como `-1` para desativar as aspas por completo.

- `Multiline` – Booleano.

  Um valor booleano que especifica se um único registro pode abranger várias linhas. Isso pode ocorrer quando um campo contém um caractere de nova linha entre aspas. Você deve definir essa opção como True (Verdadeira) se qualquer registro ocupar várias linhas. O valor padrão é `False`, que permite uma divisão de arquivos mais radical durante a análise.

- `WithHeader` – Booleano.

  Um valor booleano que especifica se é necessário tratar a primeira linha como um cabeçalho. O valor padrão é `False`.

- `WriteHeader` – Booleano.

  Um valor booleano que especifica se é necessário escrever o cabeçalho na saída. O valor padrão é `True`.

- `SkipFirst` – Booleano.

  Um valor booleano que especifica se é necessário ignorar a primeira linha de dados. O valor padrão é `False`.

- `OptimizePerformance` – Booleano.

  Um valor booleano que especifica se deve usar o leitor SIMD para CSV avançado junto com formatos de memória colunar baseados no Apache Arrow. Disponível somente na AWS Glue versão 3.0.

- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte CSV do S3.

Estrutura DirectJDBCSource

Especifica a conexão direta da fonte JDBC.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da conexão da fonte JDBC.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O banco de dados da conexão da fonte JDBC.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  A tabela da conexão da fonte JDBC.
- `ConnectionName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da conexão da fonte JDBC.
- `ConnectionType` – obrigatório: string UTF-8 (valores válidos: `sqlserver` | `mysql` | `oracle` | `postgresql` | `redshift`).

  O tipo de conexão da fonte JDBC.
- `RedshiftTmpDir` – String UTF-8 correspondente a Custom string pattern #40.

  O diretório temporário da fonte JDBC do Redshift.

Estrutura S3 DirectSourceAdditionalOptions

Especifica opções de conexão adicionais para o datastore do Amazon S3.

Campos

- `BoundedSize` – Número (extenso).

  Define o limite superior para o tamanho de destino do conjunto de dados em bytes que serão processados.
- `BoundedFiles` – Número (extenso).

  Define o limite superior para o número alvo de arquivos que serão processados.

- `EnableSamplePath` – Booleano.

  Define a opção para ativar um caminho de exemplo.

- `SamplePath` – String UTF-8 correspondente a Custom string pattern #40.

  Se ativado, especifica o caminho de exemplo.

Estrutura S3 JsonSource

Especifica um datastore JSON armazenado no Amazon S3.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do datastore.

- `Paths`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de caminhos do Amazon S3 dos quais fazer a leitura.

- `CompressionType`: string UTF-8 (valores válidos: `gzip="GZIP"` | `bzip2="BZIP2"`).

  Especifica como os dados são compactados. Isso geralmente não é necessário se os dados tem uma extensão de arquivo padrão. Os possíveis valores são `"gzip"` e `"bzip"`).

- `Exclusions` – Uma matriz de strings UTF-8.

  Uma string contendo uma lista JSON de padrões glob a excluir estilo Unix. Por exemplo, "[\"**.pdf \"]" exclui todos os arquivos PDF.

- `GroupSize` – String UTF-8 correspondente a Custom string pattern #40.

  O tamanho do grupo de destino em bytes. O padrão é calculado com base no tamanho de dados de entrada e o tamanho de seu cluster. Quando há menos de 50.000 arquivos de entrada, `"groupFiles"` deve ser definido como `"inPartition"` para poder entrar em vigor.

- `GroupFiles` – String UTF-8 correspondente a Custom string pattern #40.

  O agrupamento de arquivos é ativado por padrão quando a entrada contiver mais de 50.000 arquivos. Para habilitar o agrupamento com menos de 50.000 arquivos, defina esse parâmetro como "inPartition". Para desabilitar o agrupamento quando houver mais de 50.000 arquivos, defina esse parâmetro como `"none"`.

- `Recurse` – Booleano.

  Se definido como verdadeiro, recursivamente lê arquivos em todos os subdiretórios de acordo com os caminhos especificados.

- `MaxBand` – Número (inteiro), não mais do que None (Nenhum).

  Esta opção controla a duração, em milissegundos, após a qual a listagem do s3 provavelmente será consistente. Arquivos com registros de data e hora de modificação dentro dos últimos milissegundos do MaxBand são rastreados especialmente quando usados JobBookmarks para contabilizar a consistência eventual do Amazon S3. A maioria dos usuários não precisa definir essa opção. O valor padrão é 900.000 milissegundos, ou 15 minutos.

- `MaxFilesInBand` – Número (inteiro), não mais do que None (Nenhum).

  Esta opção especifica o número máximo de arquivos para salvar nos últimos maxBand segundos. Se esse número for excedido, os arquivos extras são ignorados e apenas processados na próxima execução do trabalho.

- `AdditionalOptions` – Um objeto S3 DirectSourceAdditionalOptions.

  Especifica opções de conexão adicionais.

- `JsonPath` – String UTF-8 correspondente a Custom string pattern #40.

  Uma JsonPath string definindo os dados JSON.

- `Multiline` – Booleano.

  Um valor booleano que especifica se um único registro pode abranger várias linhas. Isso pode ocorrer quando um campo contém um caractere de nova linha entre aspas. Você deve definir essa opção como True (Verdadeira) se qualquer registro ocupar várias linhas. O valor padrão é `False`, que permite uma divisão de arquivos mais radical durante a análise.

- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte JSON do S3.

Estrutura S3 ParquetSource

Especifica um datastore do Apache Parquet armazenado no Amazon S3.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do datastore.

- `Paths`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de caminhos do Amazon S3 dos quais fazer a leitura.

- `CompressionType` – String UTF-8 (valores válidos: `snappy="SNAPPY"` | `lzo="LZO"` | `gzip="GZIP"` | `uncompressed="UNCOMPRESSED"` | `none="NONE"`).

  Especifica como os dados são compactados. Isso geralmente não é necessário se os dados tem uma extensão de arquivo padrão. Os possíveis valores são `"gzip"` e `"bzip"`).

- `Exclusions` – Uma matriz de strings UTF-8.

  Uma string contendo uma lista JSON de padrões glob a excluir estilo Unix. Por exemplo, "[\"**.pdf \"]" exclui todos os arquivos PDF.

- `GroupSize` – String UTF-8 correspondente a Custom string pattern #40.

  O tamanho do grupo de destino em bytes. O padrão é calculado com base no tamanho de dados de entrada e o tamanho de seu cluster. Quando há menos de 50.000 arquivos de entrada, `"groupFiles"` deve ser definido como `"inPartition"` para poder entrar em vigor.

- `GroupFiles` – String UTF-8 correspondente a Custom string pattern #40.

  O agrupamento de arquivos é ativado por padrão quando a entrada contiver mais de 50.000 arquivos. Para habilitar o agrupamento com menos de 50.000 arquivos, defina esse parâmetro como "inPartition". Para desabilitar o agrupamento quando houver mais de 50.000 arquivos, defina esse parâmetro como `"none"`.

- `Recurse` – Booleano.

  Se definido como verdadeiro, recursivamente lê arquivos em todos os subdiretórios de acordo com os caminhos especificados.

- `MaxBand` – Número (inteiro), não mais do que None (Nenhum).

  Esta opção controla a duração, em milissegundos, após a qual a listagem do s3 provavelmente será consistente. Arquivos com registros de data e hora de modificação dentro dos últimos milissegundos do MaxBand são rastreados especialmente quando usados JobBookmarks para

contabilizar a consistência eventual do Amazon S3. A maioria dos usuários não precisa definir essa opção. O valor padrão é 900.000 milissegundos, ou 15 minutos.

- `MaxFilesInBand` – Número (inteiro), não mais do que None (Nenhum).

  Esta opção especifica o número máximo de arquivos para salvar nos últimos maxBand segundos. Se esse número for excedido, os arquivos extras são ignorados e apenas processados na próxima execução do trabalho.

- `AdditionalOptions` – Um objeto S3 DirectSourceAdditionalOptions.

  Especifica opções de conexão adicionais.

- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte Parquet do S3.

Estrutura S3 DeltaSource

Especifica uma fonte de dados do Delta Lake armazenada em Amazon S3.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte do Delta Lake.

- `Paths`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de caminhos do Amazon S3 dos quais fazer a leitura.

- `AdditionalDeltaOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Especifica opções de conexão adicionais.

- `AdditionalOptions` – Um objeto S3 DirectSourceAdditionalOptions.

  Especifica opções adicionais para o conector.

- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte do Delta Lake.

Estrutura S3 CatalogDeltaSource

Especifica uma fonte de dados do Delta Lake que está registrada no Catálogo AWS Glue de Dados. A fonte de dados deve ser armazenada em Amazon S3.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados do Delta Lake.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.
- `AdditionalDeltaOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Especifica opções de conexão adicionais.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte do Delta Lake.

CatalogDeltaSource estrutura

Especifica uma fonte de dados do Delta Lake que está registrada no Catálogo AWS Glue de Dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados do Delta Lake.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.
- `AdditionalDeltaOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Especifica opções de conexão adicionais.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte do Delta Lake.

Estrutura S3 HudiSource

Especifica uma fonte de dados Hudi armazenada em. Amazon S3

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte Hudi.
- `Paths`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de caminhos do Amazon S3 dos quais fazer a leitura.
- `AdditionalHudiOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Especifica opções de conexão adicionais.
- `AdditionalOptions` – Um objeto S3 DirectSourceAdditionalOptions.

  Especifica opções adicionais para o conector.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte Hudi.

Estrutura S3 CatalogHudiSource

Especifica uma fonte de dados Hudi que está registrada no Catálogo de AWS Glue Dados. A fonte de dados Hudi deve ser armazenada em Amazon S3.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados Hudi.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.
- `AdditionalHudiOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Especifica opções de conexão adicionais.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte Hudi.

CatalogHudiSource estrutura

Especifica uma fonte de dados Hudi que está registrada no Catálogo de AWS Glue Dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados Hudi.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

O nome da tabela no banco de dados da qual a leitura será feita.

- `AdditionalHudiOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Especifica opções de conexão adicionais.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a fonte Hudi.

Estrutura do DynamoDB CatalogSource

Especifica uma fonte de dados do DynamoDB no catálogo de dados. AWS Glue

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados do qual a leitura será feita.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.

RelationalCatalogSource estrutura

Especifica uma fonte de dados de banco de dados relacional no AWS Glue Data Catalog.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

O nome do banco de dados do qual a leitura será feita.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados da qual a leitura será feita.

Estrutura do JDBC ConnectorTarget

Especifica um destino de dados que grava no Amazon S3 no armazenamento colunar do Apache Parquet.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `ConnectionName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da conexão associada ao conector.

- `ConnectionTable` – Obrigatório: string UTF-8, correspondente a Custom string pattern #41.

  O nome da tabela no destino dos dados.

- `ConnectorName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome de um conector que será usado.

- `ConnectionType` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O tipo de conexão, como marketplace.jdbc ou custom.jdbc, designando uma conexão com um destino de dados JDBC.

- `AdditionalOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Opções de conexão adicionais para o conector.

- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para o destino do JDBC.

SparkConnectorTarget estrutura

Especifica um destino que usa um conector Apache Spark.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `ConnectionName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome de uma conexão para um conector do Apache Spark.

- `ConnectorName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome de um conector do Apache Spark.

- `ConnectionType` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O tipo de conexão, como marketplace.spark ou custom.spark, designando uma conexão com um datastore do Apache Spark.

- `AdditionalOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Opções de conexão adicionais para o conector.

- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para o destino do Spark personalizado.

BasicCatalogTarget estrutura

Especifica um destino que usa uma tabela do Catálogo AWS Glue de Dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do seu destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O banco de dados que contém a tabela que você deseja usar como destino. Esse banco de dados já deve existir no Data Catalog.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  A tabela que define o esquema dos dados de saída. Essa tabela já deve existir no Data Catalog.

Estrutura do MySQL CatalogTarget

Especifica um destino que usa o MySQL.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados no qual gravar.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados na qual gravar.

Estrutura do PostgreSQL CatalogTarget

Especifica um destino que usa o Postgres SQL.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados no qual gravar.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados na qual gravar.

Estrutura do OracleSQL CatalogTarget

Especifica um destino que usa o Oracle SQL.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados no qual gravar.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados na qual gravar.

Estrutura Microsoft SQL ServerCatalogTarget

Especifica um destino que usa o Microsoft SQL.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados no qual gravar.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados na qual gravar.

RedshiftTarget estrutura

Especifica um destino que usa o Amazon Redshift.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados no qual gravar.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados na qual gravar.
- `RedshiftTmpDir` – String UTF-8 correspondente a Custom string pattern #40.

O caminho do Amazon S3 onde dados temporários podem ser preparados ao serem copiados do banco de dados.

- `TmpDirIAMRole` – String UTF-8 correspondente a Custom string pattern #40.

  A função do IAM com permissões.
- `UpsertRedshiftOptions` – Um objeto UpsertRedshiftTargetOptions.

  O conjunto de opções para configurar uma operação upsert ao gravar em um destino do Redshift.

AmazonRedshiftTarget estrutura

Especifica um destino do Amazon Redshift.

Campos

- `Name` – String UTF-8 correspondente a Custom string pattern #43.

  O nome do destino do Amazon Redshift.
- `Data` – Um objeto AmazonRedshiftNodeData.

  Especifica os dados do nó de destino do Amazon Redshift.
- `Inputs`: uma matriz de strings UTF-8, com não menos que 1 nem mais que 1 string.

  Os nós que são entradas para o destino de dados.

UpsertRedshiftTargetOptions estrutura

As opções para configurar uma operação upsert ao gravar em um destino do Redshift.

Campos

- `TableLocation` – String UTF-8 correspondente a Custom string pattern #40.

  A localização física da tabela do Redshift.
- `ConnectionName` – String UTF-8 correspondente a Custom string pattern #40.

  O nome da conexão a ser usada para gravar no Redshift.
- `UpsertKeys` – Uma matriz de strings UTF-8.

As chaves usadas para determinar se uma atualização ou uma inserção será executada.

Estrutura S3 CatalogTarget

Especifica um destino de dados que grava no Amazon S3 usando AWS Glue o catálogo de dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.
- `PartitionKeys` – Uma matriz de strings UTF-8.

  Especifica o particionamento nativo usando uma sequência de chaves.
- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados na qual gravar.
- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados no qual gravar.
- `SchemaChangePolicy` – Um objeto CatalogSchemaChangePolicy.

  Uma política que especifica o comportamentos de atualização do crawler.

Estrutura S3 GlueParquetTarget

Especifica um destino de dados que grava no Amazon S3 no armazenamento colunar do Apache Parquet.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

Os nós que são entradas para o destino de dados.

- `PartitionKeys` – Uma matriz de strings UTF-8.

  Especifica o particionamento nativo usando uma sequência de chaves.

- `Path` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  Um único caminho do Amazon S3 no qual gravar.

- `Compression` – String UTF-8 (valores válidos: `snappy="SNAPPY"` | `lzo="LZO"` | `gzip="GZIP"` | `uncompressed="UNCOMPRESSED"` | `none="NONE"`).

  Especifica como os dados são compactados. Isso geralmente não é necessário se os dados tem uma extensão de arquivo padrão. Os possíveis valores são `"gzip"` e `"bzip"`).

- `SchemaChangePolicy` – Um objeto DirectSchemaChangePolicy.

  Uma política que especifica o comportamentos de atualização do crawler.

CatalogSchemaChangePolicy estrutura

Uma política que especifica o comportamentos de atualização do crawler.

Campos

- `EnableUpdateCatalog` – Booleano.

  Se comportamento de atualização especificado deve ser usado quando o crawler encontra um esquema alterado.

- `UpdateBehavior`: string UTF-8 (valores válidos: `UPDATE_IN_DATABASE` | `LOG`).

  O comportamento de atualização quando o crawler encontra um esquema alterado.

Estrutura S3 DirectTarget

Especifica um destino de dados que grava no Amazon S3.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `PartitionKeys` – Uma matriz de strings UTF-8.

  Especifica o particionamento nativo usando uma sequência de chaves.

- `Path` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  Um único caminho do Amazon S3 no qual gravar.

- `Compression` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica como os dados são compactados. Isso geralmente não é necessário se os dados tem uma extensão de arquivo padrão. Os possíveis valores são `"gzip"` e `"bzip"`).

- `Format`: obrigatório: string UTF-8 (valores válidos: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

  Especifica o formato de saída de dados para o destino.

- `SchemaChangePolicy` – Um objeto DirectSchemaChangePolicy.

  Uma política que especifica o comportamentos de atualização do crawler.

Estrutura S3 HudiCatalogTarget

Especifica um destino que grava em uma fonte de dados Hudi no Catálogo de AWS Glue Dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `PartitionKeys` – Uma matriz de strings UTF-8.

  Especifica o particionamento nativo usando uma sequência de chaves.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados na qual gravar.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados no qual gravar.

- `AdditionalOptions`: obrigatório: uma matriz de mapa dos pares de chave-valor.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Especifica as opções de conexão adicionais para o conector.

- `SchemaChangePolicy` – Um objeto CatalogSchemaChangePolicy.

  Uma política que especifica o comportamentos de atualização do crawler.

Estrutura S3 HudiDirectTarget

Especifica um destino que grava em uma fonte de dados Hudi em. Amazon S3

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `Path` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O caminho do Amazon S3 da fonte de dados Hudi na qual gravar.

- `Compression`: obrigatório: string UTF-8 (valores válidos: `gzip="GZIP"` | `lzo="LZO"` | `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  Especifica como os dados são compactados. Isso geralmente não é necessário se os dados tem uma extensão de arquivo padrão. Os possíveis valores são `"gzip"` e `"bzip"`).

- `PartitionKeys` – Uma matriz de strings UTF-8.

Especifica o particionamento nativo usando uma sequência de chaves.

- `Format`: obrigatório: string UTF-8 (valores válidos: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

  Especifica o formato de saída de dados para o destino.

- `AdditionalOptions`: obrigatório: uma matriz de mapa dos pares de chave-valor.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Especifica as opções de conexão adicionais para o conector.

- `SchemaChangePolicy` – Um objeto DirectSchemaChangePolicy.

  Uma política que especifica o comportamentos de atualização do crawler.

Estrutura S3 DeltaCatalogTarget

Especifica um destino que grava em uma fonte de dados do Delta Lake no Catálogo AWS Glue de Dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `PartitionKeys` – Uma matriz de strings UTF-8.

  Especifica o particionamento nativo usando uma sequência de chaves.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados na qual gravar.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados no qual gravar.

- `AdditionalOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Especifica as opções de conexão adicionais para o conector.
- `SchemaChangePolicy` – Um objeto CatalogSchemaChangePolicy.

  Uma política que especifica o comportamentos de atualização do crawler.

Estrutura S3 DeltaDirectTarget

Especifica um destino que grava em uma fonte de dados do Delta Lake em Amazon S3.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.
- `PartitionKeys` – Uma matriz de strings UTF-8.

  Especifica o particionamento nativo usando uma sequência de chaves.
- `Path` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O caminho do Amazon S3 da fonte de dados do Delta Lake na qual gravar.
- `Compression` – Obrigatório: string UTF-8 (valores válidos: `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  Especifica como os dados são compactados. Isso geralmente não é necessário se os dados tem uma extensão de arquivo padrão. Os possíveis valores são `"gzip"` e `"bzip"`).
- `Format`: obrigatório: string UTF-8 (valores válidos: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

  Especifica o formato de saída de dados para o destino.
- `AdditionalOptions` – Um array de mapa dos pares de valor-chave.

Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

Especifica as opções de conexão adicionais para o conector.

- `SchemaChangePolicy` – Um objeto DirectSchemaChangePolicy.

  Uma política que especifica o comportamentos de atualização do crawler.

DirectSchemaChangePolicy estrutura

Uma política que especifica o comportamentos de atualização do crawler.

Campos

- `EnableUpdateCatalog` – Booleano.

  Se comportamento de atualização especificado deve ser usado quando o crawler encontra um esquema alterado.

- `UpdateBehavior`: string UTF-8 (valores válidos: `UPDATE_IN_DATABASE` | `LOG`).

  O comportamento de atualização quando o crawler encontra um esquema alterado.

- `Table` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica a tabela no banco de dados à qual a política de alteração de esquema se aplica.

- `Database` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica o banco de dados no qual a política de alteração de esquema se aplica.

ApplyMapping estrutura

Especifica uma transformação que mapeia chaves de propriedade de dados na fonte dos dados para chaves de propriedade de dados no destino dos dados. Você pode renomear chaves, modificar os tipos de dados para chaves e escolher quais chaves remover do conjunto de dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

O nome do nó de transformação.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  As entradas de dados identificadas por seus nomes de nós.

- `Mapping` – Obrigatório: uma matriz de objetos Mapeamento.

  Especifica o mapeamento das chaves de propriedade de dados na fonte dos dados para chaves de propriedade de dados no destino dos dados.

Estrutura Mapping

Especifica o mapeamento de chaves de propriedade de dados.

Campos

- `ToKey` – String UTF-8 correspondente a Custom string pattern #40.

  Após o mapeamento de aplicação, qual deve ser o nome da coluna. Pode ser igual a `FromPath`.

- `FromPath` – Uma matriz de strings UTF-8.

  A tabela ou coluna a ser modificada.

- `FromType` – String UTF-8 correspondente a Custom string pattern #40.

  O tipo dos dados a serem modificados.

- `ToType` – String UTF-8 correspondente a Custom string pattern #40.

  O tipo de dados para o qual os dados devem ser modificados.

- `Dropped` – Booleano.

  Se verdadeiro, a coluna será removida.

- `Children` – Uma matriz de objetos Mapeamento.

  Aplicável somente a estruturas de dados aninhadas. Se você quiser alterar a estrutura pai, mas também um de seus filhos, você pode preencher esta estrutura de dados. É também `Mapping`, mas seu `FromPath` será o `FromPath` dos pais mais o `FromPath` dessa estrutura.

  Para a parte dos filhos, suponha que você tenha a estrutura:

```
{ "FromPath": "OuterStructure", "ToKey": "OuterStructure", "ToType":
"Struct", "Dropped": false, "Chidlren": [{ "FromPath": "inner", "ToKey":
"inner", "ToType": "Double", "Dropped": false, }] }
```

É possível especificar um `Mapping` parecido com:

```
{ "FromPath": "OuterStructure", "ToKey": "OuterStructure", "ToType":
"Struct", "Dropped": false, "Chidlren": [{ "FromPath": "inner", "ToKey":
"inner", "ToType": "Double", "Dropped": false, }] }
```

SelectFields estrutura

Especifica uma transformação que escolhe as chaves de propriedade de dados que você deseja manter.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  As entradas de dados identificadas por seus nomes de nós.
- `Paths`: obrigatório: uma matriz de strings UTF-8.

  Um caminho JSON para uma variável na estrutura de dados.

DropFields estrutura

Especifica uma transformação que escolhe as chaves de propriedade de dados que você deseja descartar.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

As entradas de dados identificadas por seus nomes de nós.

- `Paths`: obrigatório: uma matriz de strings UTF-8.

  Um caminho JSON para uma variável na estrutura de dados.

RenameField estrutura

Especifica uma transformação que renomeia uma única chave de propriedade de dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  As entradas de dados identificadas por seus nomes de nós.

- `SourcePath`: obrigatório: uma matriz de strings UTF-8.

  Um caminho JSON para uma variável na estrutura de dados para os dados da fonte.

- `TargetPath`: obrigatório: uma matriz de strings UTF-8.

  Um caminho JSON para uma variável na estrutura de dados para os dados do destino.

Estrutura Spigot

Especifica uma transformação que grava amostras dos dados em um bucket do Amazon S3.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  As entradas de dados identificadas por seus nomes de nós.

- `Path` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

Um caminho no Amazon S3 onde a transformação grava um subconjunto de registros do conjunto de dados em um arquivo JSON, em um bucket do Amazon S3.

- `Topk`: número (inteiro), não mais do que 100.

  Especifica vários registros a serem gravados a partir do início do conjunto de dados.

- `Prob`: número (double), no máximo 1.

  A probabilidade (um valor decimal com um valor máximo de 1) de escolher qualquer registro. Um valor de 1 indica que cada linha lida do conjunto de dados deve ser incluída na saída de amostra.

Estrutura Join

Especifica uma transformação que une dois conjuntos de dados em um só, usando uma frase de comparação nas chaves de propriedade de dados especificadas. Você pode usar junção inner (interna), outer (externa), left (à esquerda), right (à direita), left semi (semi à esquerda) e left anti (anti à esquerda).

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 2 ou mais de 2 strings.

  As entradas de dados identificadas por seus nomes de nós.

- `JoinType` – obrigatório: string UTF-8 (valores válidos: `equijoin="EQUIJOIN"` | `left="LEFT"` | `right="RIGHT"` | `outer="OUTER"` | `leftsemi="LEFT_SEMI"` | `leftanti="LEFT_ANTI"`).

  Especifica o tipo de junção a ser executada nos conjuntos de dados.

- `Columns`: obrigatório: uma matriz de objetos JoinColumn, não menos de 2 ou mais de 2 estruturas.

  Uma lista das duas colunas a serem unidas.

JoinColumn estrutura

Especifica uma coluna a ser unida.

Campos

- `From` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  A coluna a ser unida.

- `Keys`: obrigatório: uma matriz de strings UTF-8.

  A chave da coluna a ser unida.

SplitFields estrutura

Especifica uma transformação que divide chaves de propriedade de dados em dois `DynamicFrames`. A saída é uma coleção de `DynamicFrames`: um com chaves de propriedade de dados selecionadas e outro com as chaves de propriedade de dados restantes.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  As entradas de dados identificadas por seus nomes de nós.

- `Paths`: obrigatório: uma matriz de strings UTF-8.

  Um caminho JSON para uma variável na estrutura de dados.

SelectFromCollection estrutura

Especifica uma transformação que escolhe um `DynamicFrame` de uma coleção de `DynamicFrames`. A saída é o `DynamicFrame` selecionado.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

As entradas de dados identificadas por seus nomes de nós.

- `Index` – Obrigatório: número (inteiro), não mais do que Nenhum.

  O índice do DynamicFrame a ser selecionado.

FillMissingValues estrutura

Especifica uma transformação que localiza registros no conjunto de dados que tenham valores ausentes e adiciona um novo campo com um valor determinado por imputação. O conjunto de dados de entrada é usado para treinar o modelo de machine learning que determina qual deve ser o valor ausente.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  As entradas de dados identificadas por seus nomes de nós.
- `ImputedPath` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  Um caminho JSON para uma variável na estrutura de dados para o conjunto de dados imputado.
- `FilledPath` – String UTF-8 correspondente a Custom string pattern #40.

  Um caminho JSON para uma variável na estrutura de dados para o conjunto de dados preenchido.

Estrutura Filter

Especifica uma transformação que divide um conjunto de dados em dois, com base em uma condição de filtro.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

As entradas de dados identificadas por seus nomes de nós.

- `LogicalOperator` – Obrigatório: string UTF-8 (valores válidos: `AND` | `OR`).

  O operador costumava filtrar linhas comparando o valor da chave com um valor especificado.

- `Filters` – Obrigatório: uma matriz de objetos FilterExpression.

  Especifica uma expressão de filtro.

FilterExpression estrutura

Especifica uma expressão de filtro.

Campos

- `Operation`: obrigatório: string UTF-8 (valores válidos: `EQ` | `LT` | `GT` | `LTE` | `GTE` | `REGEX` | `ISNULL`).

  O tipo de operação a ser executada na expressão.

- `Negated` – Booleano.

  Se a expressão deve ser negada.

- `Values` – Obrigatório: uma matriz de objetos FilterValue.

  Uma lista de valores de filtro.

FilterValue estrutura

Representa uma única entrada na lista de valores de uma `FilterExpression`.

Campos

- `Type` – Obrigatório: string UTF-8 (valores válidos: `COLUMNEXTRACTED` | `CONSTANT`).

  O tipo de valor do filtro.

- `Value`: obrigatório: uma matriz de strings UTF-8.

  O valor a ser associado.

CustomCode estrutura

Especifica uma transformação que usa código personalizado que você fornece para executar a transformação de dados. A saída é uma coleção de DynamicFrames.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.
- `Inputs` – Obrigatório: uma matriz de strings UTF-8, pelo menos 1 string.

  As entradas de dados identificadas por seus nomes de nós.
- `Code` – Obrigatório: string UTF-8, correspondente a Custom string pattern #35.

  O código personalizado usado para executar a transformação de dados.
- `ClassName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome definido para a classe de nó de código personalizado.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a transformação de código personalizada.

Estrutura SparkSQL

Especifica uma transformação em que você insere uma consulta de SQL usando a sintaxe do Spark SQL para transformar os dados. A saída é um único `DynamicFrame`.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.
- `Inputs` – Obrigatório: uma matriz de strings UTF-8, pelo menos 1 string.

  As entradas de dados identificadas por seus nomes de nós. Você pode associar um nome de tabela a cada nó de entrada a ser usado na consulta SQL. O nome escolhido deve atender às restrições de nomenclatura do Spark SQL.
- `SqlQuery` – Obrigatório: string UTF-8, correspondente a Custom string pattern #42.

Uma consulta SQL que deve usar a sintaxe do Spark SQL e retornar um único conjunto de dados.

- `SqlAliases` – Obrigatório: uma matriz de objetos SqlAlias.

  Uma lista de aliases. Um alias permite especificar qual nome usar no SQL para uma determinada entrada. Por exemplo, você tem uma fonte de dados chamada ""MyDataSource. Se você especificar `From` como MyDataSource e `Alias` como SqlName, em seu SQL você poderá fazer:

  ```
  select * from SqlName
  ```

  e isso obtém dados de MyDataSource.

- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a transformação do SparkSQL.

SqlAlias estrutura

Representa uma única entrada na lista de valores de `SqlAliases`.

Campos

- `From` – Obrigatório: string UTF-8, correspondente a Custom string pattern #39.

  Uma tabela ou uma coluna em uma tabela.

- `Alias` – Obrigatório: string UTF-8, correspondente a Custom string pattern #41.

  Um nome temporário dado a uma tabela ou a uma coluna em uma tabela.

DropNullFields estrutura

Especifica uma transformação que remove colunas do conjunto de dados se todos os valores na coluna forem 'null'. Por padrão, o AWS Glue Studio reconhecerá objetos nulos, mas alguns valores, como cadeias de caracteres vazias, sequências de caracteres “nulas”, números inteiros -1 ou outros espaços reservados, como zeros, não são automaticamente reconhecidos como nulos.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  As entradas de dados identificadas por seus nomes de nós.

- `NullCheckBoxList` – Um objeto NullCheckBoxList.

  Uma estrutura que representa se determinados valores são reconhecidos como valores nulos para remoção.

- `NullTextList` – Uma matriz de NullValueField objetos, não mais de 50 estruturas.

  Uma estrutura que especifica uma lista de NullValueField estruturas que representam um valor nulo personalizado, como zero ou outro valor usado como um espaço reservado nulo exclusivo para o conjunto de dados.

  A transformação `DropNullFields` remove valores nulos personalizados somente se o valor do espaço reservado nulo e o tipo de dados corresponderem aos dados.

NullCheckBoxList estrutura

Representa se determinados valores são reconhecidos como valores nulos para remoção.

Campos

- `IsEmpty` – Booleano.

  Especifica que uma string vazia é considerada como um valor nulo.

- `IsNullString` – Booleano.

  Especifica que um valor com a palavra "null" é considerado como um valor nulo.

- `IsNegOne` – Booleano.

  Especifica que um valor inteiro de -1 é considerado como um valor nulo.

NullValueField estrutura

Representa um valor nulo personalizado, como zeros ou outros valores sendo usados como um espaço reservado para nulo exclusivo para o conjunto de dados.

Campos

- `Value` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O valor do espaço reservado para nulo.
- `Datatype` – Obrigatório: um objeto DataType.

  O tipo de dados do valor.

Estrutura Datatype

Uma estrutura que representa o tipo de dados do valor.

Campos

- `Id` – Obrigatório: string UTF-8, correspondente a Custom string pattern #39.

  O tipo de dados do valor.
- `Label` – Obrigatório: string UTF-8, correspondente a Custom string pattern #39.

  Um rótulo atribuído ao tipo de dados.

Estrutura Merge

Especifica uma transformação que mescla um `DynamicFrame` com um `DynamicFrame` de preparação, de acordo com as chaves primárias especificadas para identificar registros. Registros duplicados (com as mesmas chaves primárias) não são eliminados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 2 ou mais de 2 strings.

  As entradas de dados identificadas por seus nomes de nós.
- `Source` – Obrigatório: string UTF-8, correspondente a Custom string pattern #39.

  O `DynamicFrame` da fonte que será mesclado com um `DynamicFrame` de preparação.

- `PrimaryKeys`: obrigatório: uma matriz de strings UTF-8.

  A lista de campos de chave primária para corresponder aos registros da fonte e quadros dinâmicos de preparação.

Estrutura Union

Especifica uma transformação que combina as linhas de dois ou mais conjuntos de dados em um único resultado.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 2 ou mais de 2 strings.

  A entrada do ID do nó na transformação.
- `UnionType` – Obrigatório: string UTF-8 (valores válidos: `ALL` | `DISTINCT`).

  Indica o tipo de transformação Union.

  Especifique `ALL` para unir todas as linhas das fontes de dados às resultantes DynamicFrame. A união resultante não remove linhas duplicadas.

  Especifique `DISTINCT` para remover linhas duplicadas no resultado DynamicFrame.

Estrutura PIIDetection

Especifica uma transformação que identifica, remove ou mascara dados de PII.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  A entrada do ID do nó na transformação.

- `PiiType`: obrigatório: string UTF-8 (valores válidos: `RowAudit` | `RowMasking` | `ColumnAudit` | `ColumnMasking`).

  Indica o tipo de transformação PIIDetection.

- `EntityTypesToDetect`: obrigatório: uma matriz de strings UTF-8.

  Indica os tipos de entidades que a transformação PIIDetection identificará como dados de PII.

  As entidades do tipo PII incluem: 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_IDENTIFI

- `OutputColumnName` – String UTF-8 correspondente a Custom string pattern #40.

  Indica o nome da coluna de saída que conterá qualquer tipo de entidade detectado nessa linha.

- `SampleFraction`: número (double), no máximo 1.

  Indica a fração dos dados a serem amostrados ao verificar entidades de PII.

- `ThresholdFraction`: número (double), no máximo 1.

  Indica a fração dos dados que devem ser atendidos para que uma coluna seja identificada como dados de PII.

- `MaskValue`: string UTF-8, com não mais que 256 bytes de comprimento, correspondente a Custom string pattern #37.

  Indica o valor que substituirá a entidade detectada.

Estrutura Aggregate

Especifica uma transformação que agrupa linhas por campos escolhidos e calcula o valor agregado por função especificada.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Especifica os campos e linhas a serem usados como entradas para a transformação agregada.
- `Groups`: obrigatório: uma matriz de strings UTF-8.

  Especifica os campos a serem agrupados.
- `Aggs`: obrigatório: uma matriz de objetos AggregateOperation, com, no mínimo, 1 e, no máximo, 30 estruturas.

  Especifica as funções agregadas a serem executadas em campos especificados.

DropDuplicates estrutura

Especifica uma transformação que remove linhas de dados repetidos de um conjunto de dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó de transformação.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  As entradas de dados identificadas por seus nomes de nós.
- `Columns` – Uma matriz de strings UTF-8.

  O nome das colunas a serem mescladas ou removidas caso sejam repetidas.

GovernedCatalogTarget estrutura

Especifica um destino de dados que grava no Amazon S3 usando AWS Glue o catálogo de dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino de dados.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são entradas para o destino de dados.

- `PartitionKeys` – Uma matriz de strings UTF-8.

  Especifica o particionamento nativo usando uma sequência de chaves.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome da tabela no banco de dados na qual gravar.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O nome do banco de dados no qual gravar.

- `SchemaChangePolicy` – Um objeto CatalogSchemaChangePolicy.

  Uma política que especifica o comportamento do catálogo governado.

GovernedCatalogSource estrutura

Especifica o armazenamento de dados no Catálogo de AWS Glue Dados controlado.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do datastore.

- `Database` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O banco de dados do qual a leitura será feita.

- `Table` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  A tabela do banco de dados da qual a leitura será feita.

- `PartitionPredicate` – String UTF-8 correspondente a Custom string pattern #40.

  As partições que satisfazem a esse predicado são excluídas. Os arquivos dentro do período de retenção nessas partições não são excluídos. Definido como `""`, vazio por padrão.

- `AdditionalOptions` – Um objeto S3 SourceAdditionalOptions.

  Especifica opções de conexão adicionais.

AggregateOperation estrutura

Especifica o conjunto de parâmetros necessários para realizar agregação na transformação dinâmica.

Campos

- `Column`: obrigatório: uma matriz de strings UTF-8.

  Especifica a coluna no conjunto de dados em que a função de agregação será aplicada.
- `AggFunc`: obrigatório: string UTF-8 (valores válidos: `avg` | `countDistinct` | `count` | `first` | `last` | `kurtosis` | `max` | `min` | `skewness` | `stddev_samp` | `stddev_pop` | `sum` | `sumDistinct` | `var_samp` | `var_pop`).

  Especifica a função de agregação a ser aplicada.

  As possíveis funções de agregação incluem: avg countDistinct, count, first, last, kurtosis, max, min, skewness, stddev_samp, stddev_pop, sum, sumDistinct, var_samp, var_pop

GlueSchema estrutura

Especifica um esquema definido pelo usuário quando um esquema não pode ser determinado pelo AWS Glue.

Campos

- `Columns` – Uma matriz de objetos GlueStudioSchemaColumn.

  Especifica as definições de coluna que compõem um AWS Glue esquema.

GlueStudioSchemaColumn estrutura

Especifica uma única coluna em uma definição de AWS Glue esquema.

Campos

- `Name`: – Obrigatório: string UTF-8 com não mais do que 1024 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da coluna no esquema do AWS Glue Studio.

- `Type` – String UTF-8 com comprimento não superior a 131.072 bytes, correspondente a Single-line string pattern.

  O tipo de seção dessa coluna no esquema do AWS Glue Studio.

GlueStudioColumn estrutura

Especifica uma única coluna no AWS Glue Studio.

Campos

- `Key` – Obrigatório: string UTF-8, correspondente a Custom string pattern #41.

  A chave da coluna no AWS Glue Studio.

- `FullPath`: obrigatório: uma matriz de strings UTF-8.

  O URL completo da coluna no AWS Glue Studio.

- `Type` – Obrigatório: string 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"`).

  O tipo da coluna no AWS Glue Studio.

- `Children`: uma matriz de estruturas.

  Os filhos da coluna principal no AWS Glue Studio.

DynamicTransform estrutura

Especifica o conjunto de parâmetros necessários para realizar a transformação dinâmica.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  Especifica o nome da transformação dinâmica.
- `TransformName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  Especifica o nome da transformação dinâmica conforme ela aparece no editor visual do AWS Glue Studio.
- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Especifica as entradas necessárias para a transformação dinâmica.
- `Parameters` – Uma matriz de objetos TransformConfigParameter.

  Especifica os parâmetros da transformação dinâmica.
- `FunctionName` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  Especifica o nome da função da transformação dinâmica.
- `Path` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  Especifica o caminho da origem da transformação dinâmica e dos arquivos de configuração.
- `Version` – String UTF-8 correspondente a Custom string pattern #40.

  Esse campo não é usado e será removido em uma versão futura.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica o esquema de dados para a transformação dinâmica.

TransformConfigParameter estrutura

Especifica os parâmetros no arquivo de configuração da transformação dinâmica.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

Especifica o nome do parâmetro no arquivo de configuração da transformação dinâmica.

- `Type`: obrigatório: string UTF-8 (valores válidos: `str="STR"` | `int="INT"` | `float="FLOAT"` | `complex="COMPLEX"` | `bool="BOOL"` | `list="LIST"` | `null="NULL"`).

  Especifica o tipo de parâmetro no arquivo de configuração da transformação dinâmica.

- `ValidationRule` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica a regra de validação no arquivo de configuração da transformação dinâmica.

- `ValidationMessage` – String UTF-8 correspondente a Custom string pattern #40.

  Especifica a mensagem de validação no arquivo de configuração da transformação dinâmica.

- `Value` – Uma matriz de strings UTF-8.

  Especifica o valor do parâmetro no arquivo de configuração da transformação dinâmica.

- `ListType` – String UTF-8 (valores válidos: `str="STR"` | `int="INT"` | `float="FLOAT"` | `complex="COMPLEX"` | `bool="BOOL"` | `list="LIST"` | `null="NULL"`).

  Especifica o tipo de lista do parâmetro no arquivo de configuração da transformação dinâmica.

- `IsOptional` – Booleano.

  Especifica se o parâmetro é opcional ou não no arquivo de configuração da transformação dinâmica.

EvaluateDataQuality estrutura

Especifica os critérios da avaliação de qualidade dos dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da avaliação de qualidade dos dados.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  As entradas da avaliação de qualidade dos dados.

- `Ruleset`: obrigatório: string UTF-8, não menos que 1 ou mais que 65.536 bytes de comprimento, correspondente ao Custom string pattern #38.

O conjunto de regras para a avaliação de qualidade dos dados.

- `Output`: string UTF-8 (valores válidos: `PrimaryInput` | `EvaluationResults`).

  As resultado da avaliação de qualidade dos dados.

- `PublishingOptions` – Um objeto DQ ResultsPublishingOptions.

  Opções para configurar como os resultados são publicados.

- `StopJobOnFailureOptions` – Um objeto DQ StopJobOnFailureOptions.

  Opções para configurar como o trabalho será interrompido se a avaliação de qualidade dos dados falhar.

Estrutura DQ ResultsPublishingOptions

Opções para configurar como os resultados da avaliação de qualidade dos dados são publicados.

Campos

- `EvaluationContext` – String UTF-8 correspondente a Custom string pattern #39.

  O contexto da avaliação.

- `ResultsS3Prefix` – String UTF-8 correspondente a Custom string pattern #40.

  O prefixo do Amazon S3 adicionado aos resultados.

- `CloudWatchMetricsEnabled` – Booleano.

  Habilitar métricas para os resultados de qualidade dos dados.

- `ResultsPublishingEnabled` – Booleano.

  Habilitar a publicação dos resultados de qualidade dos dados.

Estrutura DQ StopJobOnFailureOptions

Opções para configurar como o trabalho será interrompido se a avaliação de qualidade dos dados falhar.

Campos

- `StopJobOnFailureTiming`: string UTF-8 (valores válidos: `Immediate` | `AfterDataLoad`).

  Quando interromper o trabalho se a avaliação de qualidade dos dados falhar. As opções são imediatas ou AfterDataLoad.

EvaluateDataQualityMultiFrame estrutura

Especifica os critérios da avaliação de qualidade dos dados.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da avaliação de qualidade dos dados.
- `Inputs` – Obrigatório: uma matriz de strings UTF-8, pelo menos 1 string.

  As entradas da avaliação de qualidade dos dados. A primeira entrada nessa lista é a fonte de dados primária.
- `AdditionalDataSources` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #43.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Os aliases de todas as fontes de dados, exceto a primária.
- `Ruleset`: obrigatório: string UTF-8, não menos que 1 ou mais que 65.536 bytes de comprimento, correspondente ao Custom string pattern #38.

  O conjunto de regras para a avaliação de qualidade dos dados.
- `PublishingOptions` – Um objeto DQ ResultsPublishingOptions.

  Opções para configurar como os resultados são publicados.
- `AdditionalOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 (valores válidos: `performanceTuning.caching="CacheOption"` | `observations.scope="ObservationsOption"`).

Cada valor é uma sequência de caracteres UTF-8.

Opções para configurar o comportamento do runtime da transformação.

- `StopJobOnFailureOptions` – Um objeto DQ StopJobOnFailureOptions.

  Opções para configurar como o trabalho será interrompido se a avaliação de qualidade dos dados falhar.

Estrutura da fórmula

Um nó do AWS Glue Studio que usa uma AWS Glue DataBrew receita em AWS Glue trabalhos.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do nó do AWS Glue Studio.

- `Inputs`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 1 strings.

  Os nós que são inseridos no nó da fórmula, identificados por ID.

- `RecipeReference` – Obrigatório: um objeto RecipeReference.

  Uma referência à DataBrew receita usada pelo nó.

RecipeReference estrutura

Uma referência a uma AWS Glue DataBrew receita.

Campos

- `RecipeArn` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  O ARN da receita. DataBrew

- `RecipeVersion` - obrigatório: string UTF-8, com não menos do que 1 nem mais do que 16 bytes de comprimento.

  O RecipeVersion da DataBrew receita.

SnowflakeNodeData estrutura

Especifica a configuração dos nós do Snowflake no Studio. AWS Glue

Campos

- `SourceType` – String UTF-8 correspondente a Custom string pattern #39.

  Especifica como os dados recuperados são especificados. Valores válidos: `"table"`, `"query"`.
- `Connection` – Um objeto Opção.

  Especifica uma conexão do catálogo AWS Glue de dados com um endpoint do Snowflake.
- `Schema` – String UTF-8.

  Especifica um esquema de banco de dados do Snowflake para seu nó usar.
- `Table` – String UTF-8.

  Especifica uma tabela do Snowflake para seu nó usar.
- `Database` – String UTF-8.

  Especifica um banco de dados do Snowflake para seu nó usar.
- `TempDir` – String UTF-8 correspondente a Custom string pattern #40.

  Não utilizado no momento.
- `IamRole` – Um objeto Opção.

  Não utilizado no momento.
- `AdditionalOptions` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #40.

  Cada valor é uma string UTF-8 que corresponde a Custom string pattern #40.

  Especifica opções adicionais passadas ao conector do Snowflake. Se as opções forem especificadas em outro lugar neste nó, isso terá precedência.
- `SampleQuery` – String UTF-8.

  Uma string SQL usada para recuperar dados com o tipo de fonte `query`.
- `PreAction` – String UTF-8.

Uma string SQL executada antes que o conector do Snowflake execute suas ações padrão.

- `PostAction` – String UTF-8.

  Uma string SQL executada depois que o conector do Snowflake executa suas ações padrão.

- `Action` – String UTF-8.

  Especifica a ação a ser realizada ao gravar em uma tabela com dados preexistentes. Valores válidos: `append`, `merge`, `truncate`, `drop`.

- `Upsert` – Booleano.

  Usado quando a ação é `append`. Especifica o comportamento da resolução quando uma linha já existe. Se verdadeiro, as linhas preexistentes serão atualizadas. Se falso, essas linhas serão inseridas.

- `MergeAction` – String UTF-8 correspondente a Custom string pattern #39.

  Especifica uma ação de mesclagem. Valores válidos: `simple`, `custom`. Se for simples, o comportamento de mesclagem será definido por `MergeWhenMatched` e `MergeWhenNotMatched`. Se for personalizado, será definido por `MergeClause`.

- `MergeWhenMatched` – String UTF-8 correspondente a Custom string pattern #39.

  Especifica como resolver registros que correspondam a dados preexistentes durante a mesclagem. Valores válidos: `update`, `delete`.

- `MergeWhenNotMatched` – String UTF-8 correspondente a Custom string pattern #39.

  Especifica como processar registros que não correspondem a dados preexistentes durante a mesclagem. Valores válidos: `insert`, `none`.

- `MergeClause` – String UTF-8.

  Uma instrução SQL que especifica um comportamento de mesclagem personalizado.

- `StagingTable` – String UTF-8.

  O nome de uma tabela de preparação usada ao executar `merge` ou fazer o upsert das ações `append`. Os dados são gravados nessa tabela e, em seguida, movidos para a `table` por uma pós-ação gerada.

- `SelectedColumns` – Uma matriz de objetos Opção.

Especifica as colunas combinadas para identificar um registro ao detectar correspondências para mesclagens e upserts. Uma lista de estruturas com as chaves `value`, `label` e `description`. Cada estrutura descreve uma coluna.

- `AutoPushdown` – Booleano.

  Especifica se o pushdown de consultas está habilitado. Se o pushdown estiver habilitado, quando uma consulta for executada no Spark, se for possível fazer pushdown de parte da consulta para o servidor do Snowflake, isso ocorrerá. Isso melhora a performance de algumas consultas.
- `TableSchema` – Uma matriz de objetos Opção.

  Define manualmente o esquema de destino para o nó. Uma lista de estruturas com as chaves `value`, `label` e `description`. Cada estrutura define uma coluna.

SnowflakeSource estrutura

Especifica uma fonte de dados do Snowflake.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome da fonte de dados do Snowflake.
- `Data` – Obrigatório: um objeto SnowflakeNodeData.

  Configuração da fonte de dados do Snowflake.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

  Especifica esquemas definidos pelo usuário para seus dados de saída.

SnowflakeTarget estrutura

Especifica um destino do Snowflake.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome do destino do Snowflake.

- `Data` – Obrigatório: um objeto SnowflakeNodeData.

  Especifica os dados do nó de destino do Snowflake.
- `Inputs`: uma matriz de strings UTF-8, com não menos que 1 nem mais que 1 string.

  Os nós que são entradas para o destino de dados.

ConnectorDataSource estrutura

Especifica uma fonte gerada com opções de conexão padrão.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome desse nó de origem.
- `ConnectionType` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  `OconnectionType`, conforme fornecido à AWS Glue biblioteca subjacente. Esse tipo de nó é compatível com os seguintes tipos de conexão:
  - `opensearch`
  - `azuresql`
  - `azurecosmos`
  - `bigquery`
  - `saphana`
  - `teradata`
  - `vertica`
- `Data`: obrigatório: uma matriz de mapa dos pares de chave-valor.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Um mapa que especifica as opções de conexão para o nó. Você pode encontrar opções de conexão padrão para o tipo de conexão correspondente na seção Parâmetros de conexão da AWS Glue documentação.
- `OutputSchemas` – Uma matriz de objetos GlueSchema.

Especifica o esquema de dados para esta fonte.

ConnectorDataTarget estrutura

Especifica um destino gerado com opções de conexão padrão.

Campos

- `Name` – Obrigatório: string UTF-8, correspondente a Custom string pattern #43.

  O nome desse nó de destino.

- `ConnectionType` – Obrigatório: string UTF-8, correspondente a Custom string pattern #40.

  OconnectionType, conforme fornecido à AWS Glue biblioteca subjacente. Esse tipo de nó é compatível com os seguintes tipos de conexão:

  - `opensearch`
  - `azuresql`
  - `azurecosmos`
  - `bigquery`
  - `saphana`
  - `teradata`
  - `vertica`

- `Data`: obrigatório: uma matriz de mapa dos pares de chave-valor.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Um mapa que especifica as opções de conexão para o nó. Você pode encontrar opções de conexão padrão para o tipo de conexão correspondente na seção Parâmetros de conexão da AWS Glue documentação.

- `Inputs`: uma matriz de strings UTF-8, com não menos que 1 nem mais que 1 string.

  Os nós que são entradas para o destino de dados.

API de trabalhos

A API de trabalhos descreve os tipos de dados de trabalhos e contém APIs para operar com trabalhos, execuções de trabalhos e acionadores no AWS Glue.

Tópicos

Tarefas

A API Jobs descreve os tipos de dados e a API relacionados à criação, atualização, exclusão ou visualização de trabalhos em AWS Glue.

Tipos de dados

Estrutura Job

Especifica uma definição de trabalho.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome que você atribui a esta definição de trabalho.
- `JobMode` – String UTF-8 (valores válidos: `SCRIPT=""` | `VISUAL=""` | `NOTEBOOK=""`).

Um modo que descreve como um trabalho foi criado. Os valores válidos são:

- `SCRIPT`- O trabalho foi criado usando o editor de scripts do AWS Glue Studio.
- `VISUAL`- O trabalho foi criado usando o editor visual do AWS Glue Studio.
- `NOTEBOOK`: o trabalho foi criado usando um caderno de sessões interativas.

Quando o campo `JobMode` está ausente ou é nulo, `SCRIPT` é atribuído como o valor padrão.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do trabalho.

- `LogUri` – String UTF-8.

  Este campo está reservado para uso futuro.

- `Role` – String UTF-8.

  O nome ou o nome de recurso da Amazon (ARN) da função do IAM associada a esta trabalho.

- `CreatedOn` – Timestamp.

  A hora e a data em que esta definição de trabalho foi criada.

- `LastModifiedOn` – Timestamp.

  O último momento em que esta definição de trabalho foi modificada.

- `ExecutionProperty` – Um objeto ExecutionProperty.

  Um `ExecutionProperty` que especifica o número máximo de execuções simultâneas permitidas para o trabalho.

- `Command` – Um objeto JobCommand.

  O `JobCommand` que executa esse trabalho.

- `DefaultArguments` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Os argumentos padrão para toda execução desse trabalho, especificados como pares de nome-valor.

Você pode especificar aqui argumentos que seu próprio script de execução de tarefas consome, bem como argumentos que ele AWS Glue mesmo consome.

Os argumentos do trabalho podem ser registrados em log. Não transmita segredos em texto simples como argumentos. Recupere segredos de uma AWS Glue Conexão AWS Secrets Manager ou de outro mecanismo de gerenciamento de segredos se você quiser mantê-los dentro do Job.

Para obter informações sobre como especificar e consumir seus próprios argumentos de trabalho, consulte o tópico Chamar APIs do AWS Glue em Python no guia do desenvolvedor.

Para obter informações sobre os argumentos que você pode fornecer a esse campo ao configurar trabalhos do Spark, consulte o tópico Special Parameters Used by AWS Glue no guia do desenvolvedor.

Para obter informações sobre os argumentos que você pode fornecer a esse campo ao configurar trabalhos do Ray, consulte o tópico Using job parameters in Ray jobs no guia do desenvolvedor.

- `NonOverridableArguments` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Argumentos para esse trabalho que não são substituídos ao fornecer argumentos de trabalho na execução de um trabalho, especificados como pares de nome-valor.

- `Connections` – Um objeto ConnectionsList.

  As conexões usadas no trabalho.

- `MaxRetries` – Número (íntegro).

  O número máximo de vezes para repetir esse trabalho após uma JobRun falha.

- `AllocatedCapacity` – Número (íntegro).

  Este campo está obsoleto. Use `MaxCapacity` em vez disso.

  O número de unidades de processamento de AWS Glue dados (DPUs) alocadas para a execução desse trabalho. Você pode alocar um mínimo de 2 DPUs. O padrão é 10. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs

e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite do trabalho em minutos. Este é o tempo máximo durante o qual uma execução de trabalho pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas) para tarefas em lotes.

  Os trabalhos de streaming devem ter valores de tempo limite inferiores a 7 dias ou 10.080 minutos. Quando o valor for deixado em branco, o trabalho será reiniciado após 7 dias, caso você não tenha configurado uma janela de manutenção. Se você tiver uma janela de manutenção de configuração, ela será reiniciada durante a janela de manutenção após 7 dias.

- `MaxCapacity` – Número (duplo).

  Para trabalhos do Glue versão 1.0 ou anterior, usando o tipo de trabalhador padrão, o número de unidades de processamento de AWS Glue dados (DPUs) que podem ser alocadas quando esse trabalho é executado. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

  Para trabalhos do Glue versão 2.0 ou posterior, você não pode um `Maximum capacity`. Em vez disso, você deve especificar um `Worker type` e o `Number of workers`.

  Não defina `MaxCapacity` se estiver usando `WorkerType` e `NumberOfWorkers`.

  O valor que pode ser alocado para `MaxCapacity` depende se você está executando um trabalho de shell do Python, um trabalho de ETL do Apache Spark ou um trabalho de ETL de streaming do Apache Spark:

  - Ao especificar um trabalho de shell do Python (`JobCommand.Name="pythonshell"`), você poderá alocar 0,0625 ou 1 DPU. O padrão é 0,0625 DPU.
  - Ao especificar um trabalho de ETL do Apache Spark (`JobCommand.Name="glueetl"`) ou um trabalho de ETL de streaming do Apache Spark (`JobCommand.Name="gluestreaming"`), é possível alocar de 2 a 100 DPUs. O padrão é de 10 DPUs. Esse tipo de trabalho não pode ter uma alocação de DPU fracionada.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

O tipo de operador predefinido que é alocado quando um trabalho é executado. Aceita um valor de G.1X, G.2X, G.4X, G.8X ou G.025X para trabalhos do Spark. Aceita o valor Z.2X para trabalhos do Ray.

- Para o tipo de operador `G.1X`, cada operador é mapeado para 1 DPU (4 vCPU, 16 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
- Para o tipo de operador `G.2X`, cada operador é mapeado para 2 DPU (8 vCPU, 32 GB de memória) com disco de 128 GB (aproximadamente 77 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
- Para o tipo de operador `G.4X`, cada operador é mapeado para 4 DPU (16 vCPU, 64 GB de memória) com disco de 256 GB (aproximadamente 235 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark AWS nas seguintes regiões: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Cingapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central), Europa (Frankfurt), Europa (Irlanda) e Europa (Estocolmo).
- Para o tipo de operador `G.8X`, cada operador é mapeado para 8 DPU (32 vCPU, 128 GB de memória) com disco de 512 GB (aproximadamente 487 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark, nas mesmas AWS regiões compatíveis com o tipo de `G.4X` trabalhador.
- Para o tipo de operador `G.025X`, cada operador é mapeado para 0,25 DPU (2 vCPU, 4 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos de streaming de baixo volume. Esse tipo de trabalhador está disponível somente para trabalhos de streaming da AWS Glue versão 3.0.
- Para o tipo de operador `Z.2X`, cada operador é mapeado para 2 M-DPU (8 vCPUs, 64 GB de memória) com 128 GB de disco (aproximadamente 120 GB livres) e fornece até 8 operadores do Ray baseados no escalador automático.

- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de determinado `workerType` que são alocados quando um trabalho é executado.

- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da estrutura de `SecurityConfiguration` a ser usada com esse trabalho.

- `NotificationProperty` – Um objeto NotificationProperty.

  Especifica propriedades de configuração de uma notificação de trabalho.

- `Running` – Booleano.

  Este campo está reservado para uso futuro.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  Nas tarefas do Spark, `GlueVersion` determina as versões do Apache Spark e do Python que AWS Glue estão disponíveis em uma tarefa. A versão do Python indica a versão compatível com trabalhos do tipo Spark.

  Os trabalhos de Ray devem definir `GlueVersion` como `4.0` ou mais. Porém, as versões do Ray, do Python e das bibliotecas adicionais disponíveis no seu trabalho do Ray são determinadas pelo parâmetro `Runtime` do comando Job.

  Para obter mais informações sobre as AWS Glue versões disponíveis e as versões correspondentes do Spark e do Python, consulte a versão Glue no guia do desenvolvedor.

  Os trabalhos criados sem especificar uma versão do Glue usam como padrão o Glue 0.9.

- `CodeGenConfigurationNodes` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #39.

  Cada valor é um objeto CodeGenConfigurationNode A.

  A representação de um gráfico acíclico direcionado no qual tanto o componente visual do Glue Studio quanto a geração de código do Glue Studio são baseados.

- `ExecutionClass`: string UTF-8, inferior a 16 bytes de comprimento (valores válidos: `FLEX=""` | `STANDARD=""`).

Indica se o trabalho é executado com uma classe de execução padrão ou flexível. A classe de execução padrão é ideal para workloads sensíveis ao tempo que exigem a inicialização rápida de trabalhos e recursos dedicados.

A classe de execução flexível é adequada para trabalhos insensíveis ao tempo, cujos horários de início e conclusão podem variar.

Somente trabalhos com a AWS Glue versão 3.0 e superior e o tipo de comando `glueetl` poderão ser definidos como `ExecutionClassFLEX`. A classe de execução flexível está disponível para trabalhos do Spark.

- `SourceControlDetails` – Um objeto SourceControlDetails.

  Os detalhes de uma configuração de controle de origem para um trabalho, permitindo a sincronização de artefatos de trabalho de ou para um repositório remoto.

- `MaintenanceWindow` – String UTF-8 correspondente a Custom string pattern #30.

  Esse campo especifica um dia da semana e uma hora para uma janela de manutenção para trabalhos de streaming. AWS Glue realiza periodicamente atividades de manutenção. Durante essas janelas de manutenção, AWS Glue será necessário reiniciar seus trabalhos de streaming.

  AWS Glue reiniciará o trabalho dentro de 3 horas da janela de manutenção especificada. Por exemplo, se você configurar a janela de manutenção para segunda-feira às 10h (GMT), seus trabalhos serão reiniciados entre 10h (GMT) e 13h (GMT).

- `ProfileName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de um perfil de AWS Glue uso associado ao trabalho.

ExecutionProperty estrutura

Uma propriedade de execução de um trabalho.

Campos

- `MaxConcurrentRuns` – Número (íntegro).

O número máximo de execuções simultâneas permitidas para o trabalho. O padrão é um. Um erro será retornado quando este limite for atingido. O valor máximo que você pode especificar é controlado por um limite de serviço.

NotificationProperty estrutura

Especifica propriedades de configuração de uma notificação.

Campos

- `NotifyDelayAfter` – Número (inteiro), pelo menos 1.

  Depois que a execução de um trabalho for iniciada, o número de minutos a esperar antes de enviar uma notificação de atraso de execução de trabalho.

JobCommand estrutura

Especifica o código executado quando um trabalho é executado.

Campos

- `Name` – String UTF-8.

  O nome do comando de trabalho. Para um trabalho de ETL do Apache Spark, ele deve ser `glueetl`. Para um trabalho de shell do Python, ele deve ser `pythonshell`. Para um trabalho de ETL de streaming do Apache Spark, ele deve ser `gluestreaming`. Para um trabalho do Ray, isso deve ser `glueray`.
- `ScriptLocation`: string UTF-8, não superior a 400.000 bytes de comprimento.

  Especifica o caminho do Amazon Simple Storage Service (Amazon S3) para um script que executa um trabalho.
- `PythonVersion` – String UTF-8 correspondente a Custom string pattern #21.

  A versão de Python que está em uso para executar um trabalho de shell em Python. Os valores permitidos são 2 ou 3.
- `Runtime`: string UTF-8, com no máximo 64 bytes de comprimento, correspondendo a Custom string pattern #29.

Em trabalhos do Ray, Runtime é usado para especificar as versões do Ray, do Python e das bibliotecas adicionais disponíveis no ambiente. Esse campo não é usado em outros tipos de trabalho. Para obter os valores de ambiente de execução compatíveis, consulte Ambientes de tempo de execução Ray compatíveis no Guia do AWS Glue desenvolvedor.

ConnectionsList estrutura

Especifica as conexões usadas por um trabalho.

Campos

- `Connections` – Uma matriz de strings UTF-8.

  Uma lista das conexões usadas pelo trabalho.

JobUpdate estrutura

Especifica as informações usadas para atualizar uma definição de trabalho existente. A definição de trabalho anterior é totalmente substituída por essa informação.

Campos

- `JobMode` – String UTF-8 (valores válidos: `SCRIPT=""` | `VISUAL=""` | `NOTEBOOK=""`).

  Um modo que descreve como um trabalho foi criado. Os valores válidos são:
  - `SCRIPT`- O trabalho foi criado usando o editor de scripts do AWS Glue Studio.
  - `VISUAL`- O trabalho foi criado usando o editor visual do AWS Glue Studio.
  - `NOTEBOOK`: o trabalho foi criado usando um caderno de sessões interativas.

  Quando o campo `JobMode` está ausente ou é nulo, `SCRIPT` é atribuído como o valor padrão.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Descrição do trabalho que está sendo definido.
- `LogUri` – String UTF-8.

  Este campo está reservado para uso futuro.
- `Role` – String UTF-8.

O nome ou o nome de recurso da Amazon (ARN) da função do IAM associada a esse trabalho (obrigatório).

- `ExecutionProperty` – Um objeto ExecutionProperty.

  Um `ExecutionProperty` que especifica o número máximo de execuções simultâneas permitidas para o trabalho.

- `Command` – Um objeto JobCommand.

  O `JobCommand` que executa esse trabalho (obrigatório).

- `DefaultArguments` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Os argumentos padrão para toda execução desse trabalho, especificados como pares de nome-valor.

  Você pode especificar aqui argumentos que seu próprio script de execução de tarefas consome, bem como argumentos que ele AWS Glue mesmo consome.

  Os argumentos do trabalho podem ser registrados em log. Não transmita segredos em texto simples como argumentos. Recupere segredos de uma AWS Glue Conexão AWS Secrets Manager ou de outro mecanismo de gerenciamento de segredos se você quiser mantê-los dentro do Job.

  Para obter informações sobre como especificar e consumir seus próprios argumentos de trabalho, consulte o tópico Chamar APIs do AWS Glue em Python no guia do desenvolvedor.

  Para obter informações sobre os argumentos que você pode fornecer a esse campo ao configurar trabalhos do Spark, consulte o tópico Special Parameters Used by AWS Glue no guia do desenvolvedor.

  Para obter informações sobre os argumentos que você pode fornecer a esse campo ao configurar trabalhos do Ray, consulte o tópico Using job parameters in Ray jobs no guia do desenvolvedor.

- `NonOverridableArguments` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma sequência de caracteres UTF-8.

Cada valor é uma sequência de caracteres UTF-8.

Argumentos para esse trabalho que não são substituídos ao fornecer argumentos de trabalho na execução de um trabalho, especificados como pares de nome-valor.

- `Connections` – Um objeto ConnectionsList.

  As conexões usadas no trabalho.

- `MaxRetries` – Número (íntegro).

  O número máximo de novas tentativas desse trabalho em caso de falha.

- `AllocatedCapacity` – Número (íntegro).

  Este campo está obsoleto. Use `MaxCapacity` em vez disso.

  O número de unidades de processamento de AWS Glue dados (DPUs) a serem alocadas para esse trabalho. Você pode alocar um mínimo de 2 DPUs. O padrão é 10. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite do trabalho em minutos. Este é o tempo máximo durante o qual uma execução de trabalho pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas) para tarefas em lotes.

  Os trabalhos de streaming devem ter valores de tempo limite inferiores a 7 dias ou 10.080 minutos. Quando o valor for deixado em branco, o trabalho será reiniciado após 7 dias, caso você não tenha configurado uma janela de manutenção. Se você tiver uma janela de manutenção de configuração, ela será reiniciada durante a janela de manutenção após 7 dias.

- `MaxCapacity` – Número (duplo).

  Para trabalhos do Glue versão 1.0 ou anterior, usando o tipo de trabalhador padrão, o número de unidades de processamento de AWS Glue dados (DPUs) que podem ser alocadas quando esse trabalho é executado. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

Para trabalhos do Glue versão 2.0+, você não pode especificar uma `Maximum capacity`. Em vez disso, você deve especificar um `Worker type` e o `Number of workers`.

Não defina `MaxCapacity` se estiver usando `WorkerType` e `NumberOfWorkers`.

O valor que pode ser alocado para `MaxCapacity` depende se você está executando um trabalho de shell do Python, um trabalho de ETL do Apache Spark ou um trabalho de ETL de streaming do Apache Spark:

- Ao especificar um trabalho de shell do Python (`JobCommand.Name`="pythonshell"), você poderá alocar 0,0625 ou 1 DPU. O padrão é 0,0625 DPU.
- Ao especificar um trabalho de ETL do Apache Spark (`JobCommand.Name`="glueetl") ou um trabalho de ETL de streaming do Apache Spark (`JobCommand.Name`="gluestreaming"), é possível alocar de 2 a 100 DPUs. O padrão é de 10 DPUs. Esse tipo de trabalho não pode ter uma alocação de DPU fracionada.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido que é alocado quando um trabalho é executado. Aceita um valor de G.1X, G.2X, G.4X, G.8X ou G.025X para trabalhos do Spark. Aceita o valor Z.2X para trabalhos do Ray.

  - Para o tipo de operador `G.1X`, cada operador é mapeado para 1 DPU (4 vCPU, 16 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
  - Para o tipo de operador `G.2X`, cada operador é mapeado para 2 DPU (8 vCPU, 32 GB de memória) com disco de 128 GB (aproximadamente 77 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
  - Para o tipo de operador `G.4X`, cada operador é mapeado para 4 DPU (16 vCPU, 64 GB de memória) com disco de 256 GB (aproximadamente 235 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark AWS nas seguintes regiões: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA

(Oregon), Ásia-Pacífico (Cingapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central), Europa (Frankfurt), Europa (Irlanda) e Europa (Estocolmo).

- Para o tipo de operador `G.8X`, cada operador é mapeado para 8 DPU (32 vCPU, 128 GB de memória) com disco de 512 GB (aproximadamente 487 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark, nas mesmas AWS regiões compatíveis com o tipo de `G.4X` trabalhador.
- Para o tipo de operador `G.025X`, cada operador é mapeado para 0,25 DPU (2 vCPU, 4 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos de streaming de baixo volume. Esse tipo de trabalhador está disponível somente para trabalhos de streaming da AWS Glue versão 3.0.
- Para o tipo de operador `Z.2X`, cada operador é mapeado para 2 M-DPU (8 vCPUs, 64 GB de memória) com 128 GB de disco (aproximadamente 120 GB livres) e fornece até 8 operadores do Ray baseados no escalador automático.

- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de determinado `workerType` que são alocados quando um trabalho é executado.

- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da estrutura de `SecurityConfiguration` a ser usada com esse trabalho.

- `NotificationProperty` – Um objeto NotificationProperty.

  Especifica as propriedades de configuração de uma notificação de trabalho.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  Nas tarefas do Spark, `GlueVersion` determina as versões do Apache Spark e do Python que AWS Glue estão disponíveis em uma tarefa. A versão do Python indica a versão compatível com trabalhos do tipo Spark.

  Os trabalhos de Ray devem definir `GlueVersion` como `4.0` ou mais. Porém, as versões do Ray, do Python e das bibliotecas adicionais disponíveis no seu trabalho do Ray são determinadas pelo parâmetro `Runtime` do comando Job.

Para obter mais informações sobre as AWS Glue versões disponíveis e as versões correspondentes do Spark e do Python, consulte a versão Glue no guia do desenvolvedor.

Os trabalhos criados sem especificar uma versão do Glue usam como padrão o Glue 0.9.

- `CodeGenConfigurationNodes` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #39.

  Cada valor é um objeto CodeGenConfigurationNode A.

  A representação de um gráfico acíclico direcionado no qual tanto o componente visual do Glue Studio quanto a geração de código do Glue Studio são baseados.

- `ExecutionClass`: string UTF-8, inferior a 16 bytes de comprimento (valores válidos: `FLEX=""` | `STANDARD=""`).

  Indica se o trabalho é executado com uma classe de execução padrão ou flexível. A classe de execução padrão é ideal para workloads sensíveis ao tempo que exigem a inicialização rápida de trabalhos e recursos dedicados.

  A classe de execução flexível é adequada para trabalhos insensíveis ao tempo, cujos horários de início e conclusão podem variar.

  Somente trabalhos com a AWS Glue versão 3.0 e superior e o tipo de comando `glueetl` poderão ser definidos como `ExecutionClassFLEX`. A classe de execução flexível está disponível para trabalhos do Spark.

- `SourceControlDetails` – Um objeto SourceControlDetails.

  Os detalhes de uma configuração de controle de origem para um trabalho, permitindo a sincronização de artefatos de trabalho de ou para um repositório remoto.

- `MaintenanceWindow` – String UTF-8 correspondente a Custom string pattern #30.

  Esse campo especifica um dia da semana e uma hora para uma janela de manutenção para trabalhos de streaming. AWS Glue realiza periodicamente atividades de manutenção. Durante essas janelas de manutenção, AWS Glue será necessário reiniciar seus trabalhos de streaming.

  AWS Glue reiniciará o trabalho dentro de 3 horas da janela de manutenção especificada. Por exemplo, se você configurar a janela de manutenção para segunda-feira às 10h (GMT), seus trabalhos serão reiniciados entre 10h (GMT) e 13h (GMT).

- `ProfileName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de um perfil de AWS Glue uso associado ao trabalho.

SourceControlDetails estrutura

Os detalhes de uma configuração de controle de origem para um trabalho, permitindo a sincronização de artefatos de trabalho de ou para um repositório remoto.

Campos

- `Provider` – String UTF-8.

  O provedor do repositório remoto.
- `Repository`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

  O nome do repositório remoto que contém os artefatos do trabalho.
- `Owner`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

  O proprietário do repositório remoto que contém os artefatos do trabalho.
- `Branch`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

  Uma ramificação opcional no repositório remoto.
- `Folder`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

  Uma pasta opcional no repositório remoto.
- `LastCommitId`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

  O último ID de confirmação no repositório remoto.
- `LastSyncTimestamp`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

  A data e a hora em que a última sincronização de trabalho foi realizada.
- `AuthStrategy` – String UTF-8.

  O tipo de autenticação, que pode ser um token de autenticação armazenado no AWS Secrets Manager ou um token de acesso pessoal.
- `AuthToken`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

O valor de um token de autorização.

Operações

- CreateJob ação (Python: create_job)
- UpdateJob ação (Python: update_job)
- GetJob ação (Python: get_job)
- GetJobs ação (Python: get_jobs)
- DeleteJob ação (Python: delete_job)
- ListJobs ação (Python: list_jobs)
- BatchGetJobs ação (Python: batch_get_jobs)

CreateJob ação (Python: create_job)

Cria uma nova definição de trabalho.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome que você atribui a esta definição de trabalho. Ele deve ser exclusivo na sua conta da .
- `JobMode` – String UTF-8 (valores válidos: `SCRIPT=""` | `VISUAL=""` | `NOTEBOOK=""`).

  Um modo que descreve como um trabalho foi criado. Os valores válidos são:
  - `SCRIPT`- O trabalho foi criado usando o editor de scripts do AWS Glue Studio.
  - `VISUAL`- O trabalho foi criado usando o editor visual do AWS Glue Studio.
  - `NOTEBOOK`: o trabalho foi criado usando um caderno de sessões interativas.

  Quando o campo `JobMode` está ausente ou é nulo, `SCRIPT` é atribuído como o valor padrão.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Descrição do trabalho que está sendo definido.
- `LogUri` – String UTF-8.

Este campo está reservado para uso futuro.

- `Role` – Obrigatório: string UTF-8.

  O nome ou o nome de recurso da Amazon (ARN) da função do IAM associada a esta trabalho.

- `ExecutionProperty` – Um objeto ExecutionProperty.

  Um `ExecutionProperty` que especifica o número máximo de execuções simultâneas permitidas para o trabalho.

- `Command` – Obrigatório: um objeto JobCommand.

  O `JobCommand` que executa esse trabalho.

- `DefaultArguments` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Os argumentos padrão para toda execução desse trabalho, especificados como pares de nome-valor.

  Você pode especificar aqui argumentos que seu próprio script de execução de tarefas consome, bem como argumentos que ele AWS Glue mesmo consome.

  Os argumentos do trabalho podem ser registrados em log. Não transmita segredos em texto simples como argumentos. Recupere segredos de uma AWS Glue Conexão AWS Secrets Manager ou de outro mecanismo de gerenciamento de segredos se você quiser mantê-los dentro do Job.

  Para obter informações sobre como especificar e consumir seus próprios argumentos de trabalho, consulte o tópico Chamar APIs do AWS Glue em Python no guia do desenvolvedor.

  Para obter informações sobre os argumentos que você pode fornecer a esse campo ao configurar trabalhos do Spark, consulte o tópico Special Parameters Used by AWS Glue no guia do desenvolvedor.

  Para obter informações sobre os argumentos que você pode fornecer a esse campo ao configurar trabalhos do Ray, consulte o tópico Using job parameters in Ray jobs no guia do desenvolvedor.

- `NonOverridableArguments` – Um array de mapa dos pares de valor-chave.

Cada chave é uma sequência de caracteres UTF-8.

Cada valor é uma sequência de caracteres UTF-8.

Argumentos para esse trabalho que não são substituídos ao fornecer argumentos de trabalho na execução de um trabalho, especificados como pares de nome-valor.

- `Connections` – Um objeto ConnectionsList.

  As conexões usadas no trabalho.

- `MaxRetries` – Número (íntegro).

  O número máximo de novas tentativas desse trabalho em caso de falha.

- `AllocatedCapacity` – Número (íntegro).

  Esse parâmetro está suspenso. Use `MaxCapacity` em vez disso.

  O número de unidades de processamento de AWS Glue dados (DPUs) a serem alocadas para esse Job. Você pode alocar um mínimo de 2 DPUs. O padrão é 10. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite do trabalho em minutos. Este é o tempo máximo durante o qual uma execução de trabalho pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas) para tarefas em lotes.

  Os trabalhos de streaming devem ter valores de tempo limite inferiores a 7 dias ou 10.080 minutos. Quando o valor for deixado em branco, o trabalho será reiniciado após 7 dias, caso você não tenha configurado uma janela de manutenção. Se você tiver uma janela de manutenção de configuração, ela será reiniciada durante a janela de manutenção após 7 dias.

- `MaxCapacity` – Número (duplo).

  Para trabalhos do Glue versão 1.0 ou anterior, usando o tipo de trabalhador padrão, o número de unidades de processamento de AWS Glue dados (DPUs) que podem ser alocadas quando esse trabalho é executado. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

Para trabalhos do Glue versão 2.0+, você não pode especificar uma `Maximum capacity`. Em vez disso, você deve especificar um `Worker type` e o `Number of workers`.

Não defina `MaxCapacity` se estiver usando `WorkerType` e `NumberOfWorkers`.

O valor que pode ser alocado para `MaxCapacity` depende se você está executando um trabalho de shell do Python, um trabalho de ETL do Apache Spark ou um trabalho de ETL de streaming do Apache Spark:

- Ao especificar um trabalho de shell do Python (`JobCommand.Name`="pythonshell"), você poderá alocar 0,0625 ou 1 DPU. O padrão é 0,0625 DPU.
- Ao especificar um trabalho de ETL do Apache Spark (`JobCommand.Name`="glueetl") ou um trabalho de ETL de streaming do Apache Spark (`JobCommand.Name`="gluestreaming"), é possível alocar de 2 a 100 DPUs. O padrão é de 10 DPUs. Esse tipo de trabalho não pode ter uma alocação de DPU fracionada.

- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da estrutura de `SecurityConfiguration` a ser usada com esse trabalho.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags a serem usadas com essa tarefa. Você pode usar tags para limitar o acesso à tarefa. Para obter mais informações sobre tags em AWS Glue, consulte AWS Tags AWS Glue in no guia do desenvolvedor.

- `NotificationProperty` – Um objeto NotificationProperty.

  Especifica propriedades de configuração de uma notificação de trabalho.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  Nas tarefas do Spark, `GlueVersion` determina as versões do Apache Spark e do Python que AWS Glue estão disponíveis em uma tarefa. A versão do Python indica a versão compatível com trabalhos do tipo Spark.

Os trabalhos de Ray devem definir `GlueVersion` como `4.0` ou mais. Porém, as versões do Ray, do Python e das bibliotecas adicionais disponíveis no seu trabalho do Ray são determinadas pelo parâmetro `Runtime` do comando Job.

Para obter mais informações sobre as AWS Glue versões disponíveis e as versões correspondentes do Spark e do Python, consulte a versão Glue no guia do desenvolvedor.

Os trabalhos criados sem especificar uma versão do Glue usam como padrão o Glue 0.9.

- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de determinado `workerType` que são alocados quando um trabalho é executado.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido que é alocado quando um trabalho é executado. Aceita um valor de G.1X, G.2X, G.4X, G.8X ou G.025X para trabalhos do Spark. Aceita o valor Z.2X para trabalhos do Ray.

  - Para o tipo de operador `G.1X`, cada operador é mapeado para 1 DPU (4 vCPU, 16 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
  - Para o tipo de operador `G.2X`, cada operador é mapeado para 2 DPU (8 vCPU, 32 GB de memória) com disco de 128 GB (aproximadamente 77 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
  - Para o tipo de operador `G.4X`, cada operador é mapeado para 4 DPU (16 vCPU, 64 GB de memória) com disco de 256 GB (aproximadamente 235 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark AWS nas seguintes regiões: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Cingapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central), Europa (Frankfurt), Europa (Irlanda) e Europa (Estocolmo).

  - Para o tipo de operador `G.8X`, cada operador é mapeado para 8 DPU (32 vCPU, 128 GB de memória) com disco de 512 GB (aproximadamente 487 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark, nas mesmas AWS regiões compatíveis com o tipo de `G.4X` trabalhador.
  - Para o tipo de operador `G.025X`, cada operador é mapeado para 0,25 DPU (2 vCPU, 4 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos de streaming de baixo volume. Esse tipo de trabalhador está disponível somente para trabalhos de streaming da AWS Glue versão 3.0.
  - Para o tipo de operador `Z.2X`, cada operador é mapeado para 2 M-DPU (8 vCPUs, 64 GB de memória) com 128 GB de disco (aproximadamente 120 GB livres) e fornece até 8 operadores do Ray baseados no escalador automático.
- `CodeGenConfigurationNodes` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8 que corresponde a Custom string pattern #39.

  Cada valor é um objeto CodeGenConfigurationNode A.

  A representação de um gráfico acíclico direcionado no qual tanto o componente visual do Glue Studio quanto a geração de código do Glue Studio são baseados.
- `ExecutionClass`: string UTF-8, inferior a 16 bytes de comprimento (valores válidos: `FLEX=""` | `STANDARD=""`).

  Indica se o trabalho é executado com uma classe de execução padrão ou flexível. A classe de execução padrão é ideal para workloads sensíveis ao tempo que exigem a inicialização rápida de trabalhos e recursos dedicados.

  A classe de execução flexível é adequada para trabalhos insensíveis ao tempo, cujos horários de início e conclusão podem variar.

  Somente trabalhos com a AWS Glue versão 3.0 e superior e o tipo de comando `glueetl` poderão ser definidos como `ExecutionClassFLEX`. A classe de execução flexível está disponível para trabalhos do Spark.
- `SourceControlDetails` – Um objeto SourceControlDetails.

Os detalhes de uma configuração de controle de origem para um trabalho, permitindo a sincronização de artefatos de trabalho de ou para um repositório remoto.

- `MaintenanceWindow` – String UTF-8 correspondente a Custom string pattern #30.

  Esse campo especifica um dia da semana e uma hora para uma janela de manutenção para trabalhos de streaming. AWS Glue realiza periodicamente atividades de manutenção. Durante essas janelas de manutenção, AWS Glue será necessário reiniciar seus trabalhos de streaming.

  AWS Glue reiniciará o trabalho dentro de 3 horas da janela de manutenção especificada. Por exemplo, se você configurar a janela de manutenção para segunda-feira às 10h (GMT), seus trabalhos serão reiniciados entre 10h (GMT) e 13h (GMT).

- `ProfileName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de um perfil de AWS Glue uso associado ao trabalho.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome exclusivo que foi fornecido para esta definição de trabalho.

Erros

- `InvalidInputException`
- `IdempotentParameterMismatchException`
- `AlreadyExistsException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

UpdateJob ação (Python: update_job)

Atualiza uma definição de trabalho existente. A definição de trabalho anterior é totalmente substituída por essa informação.

Solicitação

- `JobName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome da definição de trabalho a ser atualizada.
- `JobUpdate` – Obrigatório: um objeto JobUpdate.

  Especifica os valores com os quais a definição de trabalho será atualizada. A configuração não especificada é removida ou redefinida para os valores padrão.
- `ProfileName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de um perfil de AWS Glue uso associado ao trabalho.

Resposta

- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Retorna o nome da definição de trabalho atualizada.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

GetJob ação (Python: get_job)

Recupera uma definição de trabalho existente.

Solicitação

- `JobName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de trabalho a ser recuperada.

Resposta

- `Job` – Um objeto Trabalho.

  A definição de trabalho solicitada.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobs ação (Python: get_jobs)

Recupera todas as definições de trabalho atuais.

Solicitação

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo da resposta.

Resposta

- `Jobs` – Uma matriz de objetos Trabalho.

  Uma lista de definições de trabalho.

- `NextToken` – String UTF-8.

  Um token de continuação, caso algumas definições de trabalho ainda não tenham sido retornadas.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

DeleteJob ação (Python: delete_job)

Exclui uma definição de trabalho especificada. Se a definição de trabalho não for encontrada, nenhuma exceção será gerada.

Solicitação

- `JobName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de trabalho a ser excluída.

Resposta

- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de trabalho que foi excluída.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

ListJobs ação (Python: list_jobs)

Recupera os nomes de todos os recursos de trabalho nessa AWS conta ou os recursos com a tag especificada. Essa operação permite que você veja quais recursos estão disponíveis em sua conta e seus nomes.

Essa operação aceita o campo `Tags` opcional, que pode ser usado como um filtro na resposta, para que recursos com tags possam ser recuperados como um grupo. Se você optar por usar a filtragem por tags, apenas os recursos com a tag serão recuperados.

Solicitação

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma solicitação de continuação.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo de uma lista a ser retornada.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  Especifica apenas o retorno desses recursos com tags.

Resposta

- `JobNames` – Uma matriz de strings UTF-8.

  Os nomes de todos os trabalhos na conta ou os trabalhos com as tags especificadas.
- `NextToken` – String UTF-8.

  Um token de continuação, se a lista retornada não contiver a métrica mais recente disponível.

Erros

- `InvalidInputException`
- `EntityNotFoundException`

- `InternalServiceException`
- `OperationTimeoutException`

BatchGetJobs ação (Python: batch_get_jobs)

Retorna uma lista de metadados do recurso para uma lista de nomes de trabalho. Depois de chamar a operação `ListJobs`, você pode chamar essa operação para acessar os dados aos quais você recebeu permissões. Essa operação oferece suporte a todas as permissões do IAM, incluindo condições de permissão que usam tags.

Solicitação

- `JobNames`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de nomes de trabalho, que podem ser os nomes retornados da operação `ListJobs`.

Resposta

- `Jobs` – Uma matriz de objetos Trabalho.

  Uma lista de definições de trabalho.
- `JobsNotFound` – Uma matriz de strings UTF-8.

  Uma lista de nomes de trabalho não encontrados.

Erros

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Execuções de trabalhos

A API Jobs Runs descreve os tipos de dados e a API relacionados a iniciar, interromper ou visualizar execuções de tarefas e redefinir marcadores de tarefas em. AWS Glue O histórico de execução de trabalhos pode ser acessado por 90 dias para seu fluxo de trabalho e execução de trabalhos.

Tipos de dados

- JobRun estrutura
- Estrutura Predecessor
- JobBookmarkEntry estrutura
- BatchStopJobRunSuccessfulSubmission estrutura
- BatchStopJobRunError estrutura
- NotificationProperty estrutura

JobRun estrutura

Contém informações sobre uma execução de trabalho.

Campos

- `Id` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID dessa execução de trabalho.
- `Attempt` – Número (íntegro).

  O número de tentativas para execução deste trabalho.
- `PreviousRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da execução anterior deste trabalho. Por exemplo, o `JobRunId` especificado na ação `StartJobRun`.
- `TriggerName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho que iniciou esta execução de trabalho.
- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de trabalho que está sendo usada nesta execução.
- `JobMode` – String UTF-8 (valores válidos: `SCRIPT=""` | `VISUAL=""` | `NOTEBOOK=""`).

Um modo que descreve como um trabalho foi criado. Os valores válidos são:

- `SCRIPT`- O trabalho foi criado usando o editor de scripts do AWS Glue Studio.
- `VISUAL`- O trabalho foi criado usando o editor visual do AWS Glue Studio.
- `NOTEBOOK`: o trabalho foi criado usando um caderno de sessões interativas.

Quando o campo `JobMode` está ausente ou é nulo, `SCRIPT` é atribuído como o valor padrão.

- `StartedOn` – Timestamp.

  A data e a hora em que a execução deste trabalho foi iniciada.

- `LastModifiedOn` – Timestamp.

  A hora em que a execução desse trabalho foi modificada.

- `CompletedOn` – Timestamp.

  A data e a hora em que a execução desse trabalho foi concluída.

- `JobRunState`: string UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING` | `EXPIRED`).

  O estado atual da execução do trabalho. Para obter mais informações sobre os status de trabalhos que foram terminados de forma anormal, consulte Status de execução de trabalhos do AWS Glue.

- `Arguments` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Os argumentos de trabalho associados a esta execução. Para essa execução de tarefa, eles substituem os argumentos padrão definidos na própria definição de tarefa.

  Você pode especificar aqui argumentos que seu próprio script de execução de tarefas consome, bem como argumentos que ele AWS Glue mesmo consome.

  Os argumentos do trabalho podem ser registrados em log. Não transmita segredos em texto simples como argumentos. Recupere segredos de uma AWS Glue Conexão AWS Secrets Manager ou de outro mecanismo de gerenciamento de segredos se você quiser mantê-los dentro do Job.

Para obter informações sobre como especificar e consumir seus próprios argumentos de trabalho, consulte o tópico Chamar APIs do AWS Glue em Python no guia do desenvolvedor.

Para obter informações sobre os argumentos que você pode fornecer a esse campo ao configurar trabalhos do Spark, consulte o tópico Special Parameters Used by AWS Glue no guia do desenvolvedor.

Para obter informações sobre os argumentos que você pode fornecer a esse campo ao configurar trabalhos do Ray, consulte o tópico Using job parameters in Ray jobs no guia do desenvolvedor.

- `ErrorMessage` – String UTF-8.

  Uma mensagem de erro associada à execução deste trabalho.

- `PredecessorRuns` – Uma matriz de objetos Predecessor.

  Uma lista de predecessores para a execução deste trabalho.

- `AllocatedCapacity` – Número (íntegro).

  Este campo está obsoleto. Use `MaxCapacity` em vez disso.

  O número de unidades de processamento de AWS Glue dados (DPUs) alocadas para isso. JobRun É possível atribuir de 2 a 100 DPUs, e o padrão é 10. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

- `ExecutionTime` – Número (íntegro).

  A quantidade de tempo (em segundos) em que o trabalho executado consumiu recursos.

- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite de `JobRun` em minutos. Este é o tempo máximo durante o qual uma execução de trabalho pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. Esse valor substitui o valor de tempo limite definido no trabalho principal.

  Os trabalhos de streaming devem ter valores de tempo limite inferiores a 7 dias ou 10.080 minutos. Quando o valor for deixado em branco, o trabalho será reiniciado após 7 dias, caso você não tenha configurado uma janela de manutenção. Se você tiver uma janela de manutenção de configuração, ela será reiniciada durante a janela de manutenção após 7 dias.

- `MaxCapacity` – Número (duplo).

Para trabalhos do Glue versão 1.0 ou anterior, usando o tipo de trabalhador padrão, o número de unidades de processamento de AWS Glue dados (DPUs) que podem ser alocadas quando esse trabalho é executado. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

Para trabalhos do Glue versão 2.0+, você não pode especificar uma `Maximum capacity`. Em vez disso, você deve especificar um `Worker type` e o `Number of workers`.

Não defina `MaxCapacity` se estiver usando `WorkerType` e `NumberOfWorkers`.

O valor que pode ser alocado para `MaxCapacity` depende se você está executando um trabalho de shell do Python, um trabalho de ETL do Apache Spark ou um trabalho de ETL de streaming do Apache Spark:

- Ao especificar um trabalho de shell do Python (`JobCommand.Name`="pythonshell"), você poderá alocar 0,0625 ou 1 DPU. O padrão é 0,0625 DPU.
- Ao especificar um trabalho de ETL do Apache Spark (`JobCommand.Name`="glueetl") ou um trabalho de ETL de streaming do Apache Spark (`JobCommand.Name`="gluestreaming"), é possível alocar de 2 a 100 DPUs. O padrão é de 10 DPUs. Esse tipo de trabalho não pode ter uma alocação de DPU fracionada.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido que é alocado quando um trabalho é executado. Aceita um valor de G.1X, G.2X, G.4X, G.8X ou G.025X para trabalhos do Spark. Aceita o valor Z.2X para trabalhos do Ray.

  - Para o tipo de operador `G.1X`, cada operador é mapeado para 1 DPU (4 vCPU, 16 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
  - Para o tipo de operador `G.2X`, cada operador é mapeado para 2 DPU (8 vCPU, 32 GB de memória) com disco de 128 GB (aproximadamente 77 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.

  - Para o tipo de operador `G.4X`, cada operador é mapeado para 4 DPU (16 vCPU, 64 GB de memória) com disco de 256 GB (aproximadamente 235 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark AWS nas seguintes regiões: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Cingapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central), Europa (Frankfurt), Europa (Irlanda) e Europa (Estocolmo).
  - Para o tipo de operador `G.8X`, cada operador é mapeado para 8 DPU (32 vCPU, 128 GB de memória) com disco de 512 GB (aproximadamente 487 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark, nas mesmas AWS regiões compatíveis com o tipo de `G.4X` trabalhador.
  - Para o tipo de operador `G.025X`, cada operador é mapeado para 0,25 DPU (2 vCPU, 4 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos de streaming de baixo volume. Esse tipo de trabalhador está disponível somente para trabalhos de streaming da AWS Glue versão 3.0.
  - Para o tipo de operador `Z.2X`, cada operador é mapeado para 2 M-DPU (8 vCPUs, 64 GB de memória) com 128 GB de disco (aproximadamente 120 GB livres) e fornece até 8 operadores do Ray baseados no escalador automático.
- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de determinado `workerType` que são alocados quando um trabalho é executado.
- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da estrutura de `SecurityConfiguration` a ser usada com a execução desse trabalho.
- `LogGroupName` – String UTF-8.

  O nome do grupo de registros para registro seguro que pode ser criptografado no lado do servidor na Amazon usando. CloudWatch AWS KMS Esse nome pode ser `/aws-glue/jobs/` e, nesse caso, a criptografia padrão é `NONE`. Se você adicionar o nome de uma função e o nome de `SecurityConfiguration` (em outras palavras, `/aws-glue/jobs-yourRoleName-`

`yourSecurityConfigurationName/`), essa configuração de segurança será usada para criptografar o grupo de logs.

- `NotificationProperty` – Um objeto NotificationProperty.

  Especifica propriedades de configuração de uma notificação de execução de trabalho.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  Nas tarefas do Spark, `GlueVersion` determina as versões do Apache Spark e do Python que AWS Glue estão disponíveis em uma tarefa. A versão do Python indica a versão compatível com trabalhos do tipo Spark.

  Os trabalhos de Ray devem definir `GlueVersion` como `4.0` ou mais. Porém, as versões do Ray, do Python e das bibliotecas adicionais disponíveis no seu trabalho do Ray são determinadas pelo parâmetro `Runtime` do comando Job.

  Para obter mais informações sobre as AWS Glue versões disponíveis e as versões correspondentes do Spark e do Python, consulte a versão Glue no guia do desenvolvedor.

  Os trabalhos criados sem especificar uma versão do Glue usam como padrão o Glue 0.9.

- `DPUSeconds` – Número (duplo).

  Esse campo pode ser preenchido para execuções de trabalhos com classes de execução `FLEX` ou quando o Auto Scaling está habilitado e representa o tempo total de execução de um executor durante o ciclo de vida de uma execução de trabalho em segundos, multiplicado por um fator de DPU (1 para operadores `G.1X`, 2 para operadores `G.2X` e 0,25 para operadores `G.025X`). Esse valor pode ser diferente de `executionEngineRuntime` * `MaxCapacity`, como no caso de trabalhos do Auto Scaling, pois o número de executores em execução em um determinado momento pode ser menor que `MaxCapacity`. Portanto, é possível que o valor de `DPUSeconds` seja menor que `executionEngineRuntime` * `MaxCapacity`.

- `ExecutionClass`: string UTF-8, inferior a 16 bytes de comprimento (valores válidos: `FLEX=""` | `STANDARD=""`).

  Indica se o trabalho é executado com uma classe de execução padrão ou flexível. A classe de execução padrão é ideal para workloads sensíveis ao tempo que exigem a inicialização rápida de trabalhos e recursos dedicados.

A classe de execução flexível é adequada para trabalhos insensíveis ao tempo, cujos horários de início e conclusão podem variar.

Somente trabalhos com a AWS Glue versão 3.0 e superior e o tipo de comando `glueetl` poderão ser definidos como `ExecutionClassFLEX`. A classe de execução flexível está disponível para trabalhos do Spark.

- `MaintenanceWindow` – String UTF-8 correspondente a Custom string pattern #30.

  Esse campo especifica um dia da semana e uma hora para uma janela de manutenção para trabalhos de streaming. AWS Glue realiza periodicamente atividades de manutenção. Durante essas janelas de manutenção, AWS Glue será necessário reiniciar seus trabalhos de streaming.

  AWS Glue reiniciará o trabalho dentro de 3 horas da janela de manutenção especificada. Por exemplo, se você configurar a janela de manutenção para segunda-feira às 10h (GMT), seus trabalhos serão reiniciados entre 10h (GMT) e 13h (GMT).
- `ProfileName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de um perfil de AWS Glue uso associado à execução do trabalho.

Estrutura Predecessor

Uma execução de trabalho usada no predicado de um gatilho condicional que a acionou.

Campos

- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de trabalho usada pela execução do trabalho antecessor.
- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da execução de trabalho predecessor.

JobBookmarkEntry estrutura

Define um ponto em que um trabalho pode retomar o processamento.

Campos

- `JobName` – String UTF-8.

  O nome do trabalho em questão.

- `Version` – Número (íntegro).

  A versão do trabalho.

- `Run` – Número (íntegro).

  O número do ID de execução.

- `Attempt` – Número (íntegro).

  O número do ID de tentativa.

- `PreviousRunId` – String UTF-8.

  O identificador de execução exclusivo associado à execução do trabalho anterior.

- `RunId` – String UTF-8.

  O número do ID de execução.

- `JobBookmark` – String UTF-8.

  O próprio marcador.

BatchStopJobRunSuccessfulSubmission estrutura

Registra uma solicitação bem-sucedido para interromper um `JobRun` especificado.

Campos

- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de trabalho usada na execução do trabalho que foi interrompida.

- `JobRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O `JobRunId` da execução de trabalho que foi interrompida.

BatchStopJobRunError estrutura

Registra um erro que ocorreu ao tentar interromper uma execução de trabalho especificada.

Campos

- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de trabalho usada na execução do trabalho em questão.
- `JobRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O `JobRunId` da execução do trabalho em questão.
- `ErrorDetail` – Um objeto ErrorDetail.

  Especifica detalhes sobre o erro encontrado.

NotificationProperty estrutura

Especifica propriedades de configuração de uma notificação.

Campos

- `NotifyDelayAfter` – Número (inteiro), pelo menos 1.

  Depois que a execução de um trabalho for iniciada, o número de minutos a esperar antes de enviar uma notificação de atraso de execução de trabalho.

Operações

- StartJobRun ação (Python: start_job_run)
- BatchStopJobRun ação (Python: batch_stop_job_run)
- GetJobRun ação (Python: get_job_run)
- GetJobRuns ação (Python: get_job_runs)
- GetJobBookmark ação (Python: get_job_bookmark)
- GetJobBookmarks ação (Python: get_job_bookmarks)
- ResetJobBookmark ação (Python: reset_job_bookmark)

StartJobRun ação (Python: start_job_run)

Inicia uma execução de trabalho usando uma definição de trabalho.

Solicitação

- `JobName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de trabalho que será usada.

- `JobRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de um `JobRun` anterior para tentar novamente.

- `Arguments` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Os argumentos de trabalho associados a esta execução. Para essa execução de tarefa, eles substituem os argumentos padrão definidos na própria definição de tarefa.

  Você pode especificar aqui argumentos que seu próprio script de execução de tarefas consome, bem como argumentos que ele AWS Glue mesmo consome.

  Os argumentos do trabalho podem ser registrados em log. Não transmita segredos em texto simples como argumentos. Recupere segredos de uma AWS Glue Conexão AWS Secrets Manager ou de outro mecanismo de gerenciamento de segredos se você quiser mantê-los dentro do Job.

  Para obter informações sobre como especificar e consumir seus próprios argumentos de trabalho, consulte o tópico Chamar APIs do AWS Glue em Python no guia do desenvolvedor.

  Para obter informações sobre os argumentos que você pode fornecer a esse campo ao configurar trabalhos do Spark, consulte o tópico Special Parameters Used by AWS Glue no guia do desenvolvedor.

  Para obter informações sobre os argumentos que você pode fornecer a esse campo ao configurar trabalhos do Ray, consulte o tópico Using job parameters in Ray jobs no guia do desenvolvedor.

- `AllocatedCapacity` – Número (íntegro).

  Este campo está obsoleto. Use `MaxCapacity` em vez disso.

  O número de unidades de processamento de AWS Glue dados (DPUs) a serem alocadas para isso. JobRun Você pode alocar um mínimo de 2 DPUs. O padrão é 10. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite de `JobRun` em minutos. Este é o tempo máximo durante o qual uma execução de trabalho pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. Esse valor substitui o valor de tempo limite definido no trabalho principal.

  Os trabalhos de streaming devem ter valores de tempo limite inferiores a 7 dias ou 10.080 minutos. Quando o valor for deixado em branco, o trabalho será reiniciado após 7 dias, caso você não tenha configurado uma janela de manutenção. Se você tiver uma janela de manutenção de configuração, ela será reiniciada durante a janela de manutenção após 7 dias.

- `MaxCapacity` – Número (duplo).

  Para trabalhos do Glue versão 1.0 ou anterior, usando o tipo de trabalhador padrão, o número de unidades de processamento de AWS Glue dados (DPUs) que podem ser alocadas quando esse trabalho é executado. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

  Para trabalhos do Glue versão 2.0+, você não pode especificar uma `Maximum capacity`. Em vez disso, você deve especificar um `Worker type` e o `Number of workers`.

  Não defina `MaxCapacity` se estiver usando `WorkerType` e `NumberOfWorkers`.

  O valor que pode ser alocado para `MaxCapacity` depende se você está executando um trabalho de shell do Python, um trabalho de ETL do Apache Spark ou um trabalho de ETL de streaming do Apache Spark:

  - Ao especificar um trabalho de shell do Python (`JobCommand.Name="pythonshell"`), você poderá alocar 0,0625 ou 1 DPU. O padrão é 0,0625 DPU.

  - Ao especificar um trabalho de ETL do Apache Spark (`JobCommand.Name="glueetl"`) ou um trabalho de ETL de streaming do Apache Spark (`JobCommand.Name="gluestreaming"`), é possível alocar de 2 a 100 DPUs. O padrão é de 10 DPUs. Esse tipo de trabalho não pode ter uma alocação de DPU fracionada.
- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da estrutura de `SecurityConfiguration` a ser usada com a execução desse trabalho.
- `NotificationProperty` – Um objeto NotificationProperty.

  Especifica propriedades de configuração de uma notificação de execução de trabalho.
- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido que é alocado quando um trabalho é executado. Aceita um valor de G.1X, G.2X, G.4X, G.8X ou G.025X para trabalhos do Spark. Aceita o valor Z.2X para trabalhos do Ray.
  - Para o tipo de operador `G.1X`, cada operador é mapeado para 1 DPU (4 vCPU, 16 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
  - Para o tipo de operador `G.2X`, cada operador é mapeado para 2 DPU (8 vCPU, 32 GB de memória) com disco de 128 GB (aproximadamente 77 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
  - Para o tipo de operador `G.4X`, cada operador é mapeado para 4 DPU (16 vCPU, 64 GB de memória) com disco de 256 GB (aproximadamente 235 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark AWS nas seguintes regiões: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Cingapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central), Europa (Frankfurt), Europa (Irlanda) e Europa (Estocolmo).

  - Para o tipo de operador `G.8X`, cada operador é mapeado para 8 DPU (32 vCPU, 128 GB de memória) com disco de 512 GB (aproximadamente 487 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark, nas mesmas AWS regiões compatíveis com o tipo de `G.4X` trabalhador.
  - Para o tipo de operador `G.025X`, cada operador é mapeado para 0,25 DPU (2 vCPU, 4 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos de streaming de baixo volume. Esse tipo de trabalhador está disponível somente para trabalhos de streaming da AWS Glue versão 3.0.
  - Para o tipo de operador `Z.2X`, cada operador é mapeado para 2 M-DPU (8 vCPUs, 64 GB de memória) com 128 GB de disco (aproximadamente 120 GB livres) e fornece até 8 operadores do Ray baseados no escalador automático.
- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de determinado `workerType` que são alocados quando um trabalho é executado.
- `ExecutionClass`: string UTF-8, inferior a 16 bytes de comprimento (valores válidos: `FLEX=""` | `STANDARD=""`).

  Indica se o trabalho é executado com uma classe de execução padrão ou flexível. A classe de execução padrão é ideal para workloads sensíveis ao tempo que exigem a inicialização rápida de trabalhos e recursos dedicados.

  A classe de execução flexível é adequada para trabalhos insensíveis ao tempo, cujos horários de início e conclusão podem variar.

  Somente trabalhos com a AWS Glue versão 3.0 e superior e o tipo de comando `glueetl` poderão ser definidos como `ExecutionClassFLEX`. A classe de execução flexível está disponível para trabalhos do Spark.
- `ProfileName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de um perfil de AWS Glue uso associado à execução do trabalho.

Resposta

- `JobRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID atribuído a esta execução de trabalho.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

BatchStopJobRun ação (Python: batch_stop_job_run)

Interrompe uma ou mais execuções de trabalho para uma definição de trabalho determinada.

Solicitação

- `JobName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de trabalho para a qual as execuções de trabalho são interrompidas.
- `JobRunIds` – Obrigatório: uma matriz de strings UTF-8, no mínimo 1 ou mais de 25 strings.

  Uma lista dos `JobRunIds` que devem ser interrompidos para essa definição de trabalho.

Resposta

- `SuccessfulSubmissions` – Uma matriz de objetos BatchStopJobRunSuccessfulSubmission.

  Uma lista dos JobRuns que foram enviados com sucesso para serem interrompidos.
- `Errors` – Uma matriz de objetos BatchStopJobRunError.

Uma lista dos erros que foram encontrados durante a tentativa de interrupção de `JobRuns`, incluindo o `JobRunId` para o qual cada erro foi encontrado e os detalhes do erro.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobRun ação (Python: get_job_run)

Recupera os metadados para uma execução de trabalho específica. O histórico de execução de trabalhos pode ser acessado por 90 dias para seu fluxo de trabalho e execução de trabalhos.

Solicitação

- `JobName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome da definição de trabalho em execução.
- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da execução de trabalho.
- `PredecessorsIncluded` – Booleano.

  True, se uma lista de execuções antecessoras for retornada.

Resposta

- `JobRun` – Um objeto JobRun.

  Os metadados solicitados da execução de trabalho.

Erros

- `InvalidInputException`

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobRuns ação (Python: get_job_runs)

Recupera os metadados para todas as execuções de uma determinada definição de trabalho.

Solicitação

- `JobName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da definição de trabalho para a qual todas as execuções de trabalho são recuperadas.
- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.
- `MaxResults`: número (inteiro) não inferior a 1 nem superior a 200.

  O tamanho máximo da resposta.

Resposta

- `JobRuns` – Uma matriz de objetos JobRun.

  Uma lista de objetos de metadados de execução de trabalho.
- `NextToken` – String UTF-8.

  Um token de continuação, se todas as execuções de trabalho solicitadas não tiverem sido retornadas.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobBookmark ação (Python: get_job_bookmark)

Retorna informações sobre uma entrada de marcador de trabalho.

Para obter mais informações sobre como habilitar e usar marcadores de trabalho, consulte:

- Rastrear dados processados usando marcadores de trabalho
- Parâmetros de trabalho usados por AWS Glue
- Estrutura de trabalhos

Solicitação

- `JobName` – Obrigatório: string UTF-8.

  O nome do trabalho em questão.
- `Version` – Número (íntegro).

  A versão do trabalho.
- `RunId` – String UTF-8.

  O identificador de execução exclusivo associado à execução desse trabalho.

Resposta

- `JobBookmarkEntry` – Um objeto JobBookmarkEntry.

  Uma estrutura que define um ponto em que um trabalho pode retomar o processamento.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ValidationException`

GetJobBookmarks ação (Python: get_job_bookmarks)

Retorna informações sobre as entradas de marcador de trabalho. A ordem da lista é em números de versão decrescentes.

Para obter mais informações sobre como habilitar e usar marcadores de trabalho, consulte:

- Rastrear dados processados usando marcadores de trabalho
- Parâmetros de trabalho usados por AWS Glue
- Estrutura de trabalhos

Solicitação

- `JobName` – Obrigatório: string UTF-8.

  O nome do trabalho em questão.
- `MaxResults` – Número (íntegro).

  O tamanho máximo da resposta.
- `NextToken` – Número (íntegro).

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `JobBookmarkEntries` – Uma matriz de objetos JobBookmarkEntry.

  Uma lista de entradas de marcador de trabalho que define um ponto em que um trabalho pode retomar o processamento.
- `NextToken` – Número (íntegro).

  Um token de continuação, que terá um valor de 1 se todas as entradas forem retornadas ou > 1 se nem todas as execuções solicitadas forem retornadas.

Erros

- `InvalidInputException`
- `EntityNotFoundException`

- `InternalServiceException`
- `OperationTimeoutException`

ResetJobBookmark ação (Python: reset_job_bookmark)

Redefine uma entrada de marcador.

Para obter mais informações sobre como habilitar e usar marcadores de trabalho, consulte:

- Rastrear dados processados usando marcadores de trabalho
- Parâmetros de trabalho usados por AWS Glue
- Estrutura de trabalhos

Solicitação

- `JobName` – Obrigatório: string UTF-8.

  O nome do trabalho em questão.

- `RunId` – String UTF-8.

  O identificador de execução exclusivo associado à execução desse trabalho.

Resposta

- `JobBookmarkEntry` – Um objeto JobBookmarkEntry.

  A redefinição da entrada de marcador.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acionadores

A API de acionadores descreve os tipos de dados e a API relacionada à criação, atualização ou exclusão, e início e interrupção, de acionadores de trabalho no AWS Glue.

Tipos de dados

- Estrutura de acionador
- Estrutura TriggerUpdate
- Estrutura Predicate
- Estrutura Condition
- Estrutura Action
- Estrutura EventBatchingCondition

Estrutura de acionador

As informações sobre um determinado gatilho.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho.
- `WorkflowName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do fluxo de trabalho associado ao gatilho.
- `Id` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Reservado para uso futuro.
- `Type` – String UTF-8 (valores válidos: `SCHEDULED` | `CONDITIONAL` | `ON_DEMAND` | `EVENT`).

  O tipo de gatilho.
- `State` – String UTF-8 (valores válidos: `CREATING` | `CREATED` | `ACTIVATING` | `ACTIVATED` | `DEACTIVATING` | `DEACTIVATED` | `DELETING` | `UPDATING`).

O estado atual do gatilho do trabalho.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição desse gatilho.

- `Schedule` – String UTF-8.

  Uma expressão `cron` usada para especificar a programação (consulte Programações baseadas em hora para tarefas e crawlers. Por exemplo, para executar algo todos os dias às 12h15 UTC, especifique: `cron(15 12 * * ? *)`.

- `Actions` – Uma matriz de objetos Ação.

  As ações iniciadas por esse gatilho.

- `Predicate` – Um objeto Predicado.

  O predicado deste gatilho, que define quando ele disparará.

- `EventBatchingCondition` – Um objeto EventBatchingCondition.

  Condição de lote que deve ser atendida (número especificado de eventos recebidos ou janela de tempo de lote expirada) antes que o acionador do evento do EventBridge dispare.

Estrutura TriggerUpdate

Uma estrutura usada para fornecer as informações necessárias para atualizar um gatilho. Este objeto atualiza a definição do gatilho anterior, substituindo-a completamente.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Reservado para uso futuro.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição desse gatilho.

- `Schedule` – String UTF-8.

Uma expressão `cron` usada para especificar a programação (consulte Programações baseadas em hora para tarefas e crawlers. Por exemplo, para executar algo todos os dias às 12h15 UTC, especifique: `cron(15 12 * * ? *)`.

- `Actions` – Uma matriz de objetos Ação.

  As ações iniciadas por esse gatilho.

- `Predicate` – Um objeto Predicado.

  O predicado deste gatilho, que define quando ele disparará.

- `EventBatchingCondition` – Um objeto EventBatchingCondition.

  Condição de lote que deve ser atendida (número especificado de eventos recebidos ou janela de tempo de lote expirada) antes que o acionador do evento do EventBridge dispare.

Estrutura Predicate

Define o predicado do gatilho, que determina quando ele é acionado.

Campos

- `Logical`: string UTF-8 (valores válidos: `AND` | `ANY`).

  Um campo opcional se apenas uma condição estiver listada. Se várias condições estiverem listadas, esse campo será obrigatório.

- `Conditions` – Uma matriz de objetos Condição.

  Uma lista das condições que determinam quando o gatilho será acionado.

Estrutura Condition

Define em que condição um gatilho será acionado.

Campos

- `LogicalOperator` – String UTF-8 (valores válidos: `EQUALS`).

  Um operador lógico.

- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do trabalho cujo `JobRuns` essa condição se aplica e na qual esse gatilho aguardará.

- `State`: string UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING` | `EXPIRED`).

  O estado da condição. No momento, os únicos estados de trabalho que um trigger pode escutar são `SUCCEEDED`, `STOPPED`, `FAILED` e `TIMEOUT`. Os únicos estados de crawler que um trigger pode escutar são `SUCCEEDED`, `FAILED` e `CANCELLED`.

- `CrawlerName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do crawler ao qual essa condição se aplica.

- `CrawlState` – String UTF-8 (valores válidos: `RUNNING` | `CANCELLING` | `CANCELLED` | `SUCCEEDED` | `FAILED` | `ERROR`).

  O estado do crawler ao qual essa condição se aplica.

Estrutura Action

Define uma ação a ser iniciada por um gatilho.

Campos

- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de um trabalho a ser executado.

- `Arguments` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Os argumentos de tarefa usados ao acionar esse trigger. Para essa execução de tarefa, eles substituem os argumentos padrão definidos na própria definição de tarefa.

Aqui, você pode especificar os argumentos que seu próprio script de execução de trabalho consome, além dos argumentos que o AWS Glue consome.

Para obter informações sobre como especificar e consumir seus próprios argumentos de trabalho, consulte o tópico Chamar APIs do AWS Glue em Python no guia do desenvolvedor.

Para obter informações sobre os pares de chave-valor que o AWS Glue consome para configurar o trabalho, consulte o tópico Parâmetros especiais usados pelo AWS Glue no guia do desenvolvedor.

- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite de `JobRun` em minutos. Este é o tempo máximo durante o qual uma execução de trabalho pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas). Isso substitui o valor de tempo limite definido no trabalho principal.
- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da estrutura de `SecurityConfiguration` a ser usada com essa ação.
- `NotificationProperty` – Um objeto NotificationProperty.

  Especifica propriedades de configuração de uma notificação de execução de trabalho.
- `CrawlerName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do crawler que será usado com essa ação.

Estrutura EventBatchingCondition

Condição de lote que deve ser atendida (número especificado de eventos recebidos ou janela de tempo de lote expirada) antes que o acionador do evento do EventBridge dispare.

Campos

- `BatchSize`: obrigatório: número (inteiro), não menos do que 1 ou superior a 100.

  Número de eventos que devem ser recebidos do Amazon EventBridge antes que o acionador do evento do EventBridge dispare.
- `BatchWindow`: número (inteiro), não menos do que 1 ou superior a 900.

Janela de tempo em segundos após a qual o acionador do evento do EventBridge dispara. A janela é iniciada quando o primeiro evento é recebido.

Operações

- Ação CreateTrigger (Python: create_trigger)
- Ação StartTrigger (Python: start_trigger)
- Ação GetTrigger (Python: get_trigger)
- Ação GetTriggers (Python: get_triggers)
- Ação UpdateTrigger (Python: update_trigger)
- Ação StopTrigger (Python: stop_trigger)
- Ação DeleteTrigger (Python: delete_trigger)
- Ação ListTriggers (Python: list_triggers)
- Ação BatchGetTriggers (Python: batch_get_triggers)

Ação CreateTrigger (Python: create_trigger)

Cria um novo gatilho.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho.
- `WorkflowName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do fluxo de trabalho associado ao gatilho.
- `Type`: obrigatório: string UTF-8 (valores válidos: `SCHEDULED` | `CONDITIONAL` | `ON_DEMAND` | `EVENT`).

  O tipo do novo gatilho.
- `Schedule` – String UTF-8.

Uma expressão `cron` usada para especificar a programação (consulte Programações baseadas em hora para tarefas e crawlers. Por exemplo, para executar algo todos os dias às 12h15 UTC, especifique: `cron(15 12 * * ? *)`.

Esse campo é obrigatório quando o tipo do gatilho é SCHEDULED.

- `Predicate` – Um objeto Predicado.

  Um predicado para especificar quando o novo gatilho será disparado.

  Esse campo é obrigatório quando o tipo do gatilho é `CONDITIONAL`.

- `Actions` – Obrigatório: uma matriz de objetos Ação.

  As ações iniciadas por esse gatilho quando ele é disparado.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do novo gatilho.

- `StartOnCreation` – Booleano.

  Defina como `true` para iniciar os gatilhos `SCHEDULED` e `CONDITIONAL` na criação. True (Verdadeiro) não é compatível com gatilhos `ON_DEMAND`.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags a serem usadas com esse trigger. Você pode usar tags para limitar o acesso ao trigger. Para obter mais informações sobre tags no AWS Glue, consulte Tags da AWS no AWS Glue no guia do desenvolvedor.

- `EventBatchingCondition` – Um objeto EventBatchingCondition.

  Condição de lote que deve ser atendida (número especificado de eventos recebidos ou janela de tempo de lote expirada) antes que o acionador do evento do EventBridge dispare.

Resposta

- Name – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho.

Erros

- `AlreadyExistsException`
- `EntityNotFoundException`
- `InvalidInputException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

Ação StartTrigger (Python: start_trigger)

Inicia um gatilho existente. Consulte Acionar tarefas para obter informações sobre como diferentes tipos de gatilho são acionados.

Solicitação

- Name – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho a ser iniciado.

Resposta

- Name – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho iniciado.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

Ação GetTrigger (Python: get_trigger)

Recupera a definição de um gatilho.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho a ser recuperado.

Resposta

- `Trigger` – Um objeto Trigger.

  A definição de gatilho solicitada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetTriggers (Python: get_triggers)

Obtém todos os gatilhos associados a um trabalho.

Solicitação

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.
- `DependentJobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do trabalho para o qual recuperar gatilhos. O gatilho que pode iniciar esse trabalho será retornado e, se ele não existir, todos os gatilhos serão retornados.
- `MaxResults`: número (inteiro) não inferior a 1 nem superior a 200.

  O tamanho máximo da resposta.

Resposta

- `Triggers` – Uma matriz de objetos Trigger.

  Uma lista de gatilhos para o trabalho especificado.
- `NextToken` – String UTF-8.

  Um token de continuação, se todos os gatilhos solicitados ainda não tiverem sido retornados.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação UpdateTrigger (Python: update_trigger)

Atualiza uma definição do gatilho.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome do gatilho a ser atualizado.

- `TriggerUpdate` – Obrigatório: um objeto TriggerUpdate.

  Os novos valores para atualização do gatilho.

Resposta

- `Trigger` – Um objeto Trigger.

  A definição de gatilho resultante.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Ação StopTrigger (Python: stop_trigger)

Interrompe um gatilho especificado.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho a ser interrompido.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho interrompido.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Ação DeleteTrigger (Python: delete_trigger)

Exclui um gatilho especificado. Se o gatilho não for encontrado, nenhuma exceção será lançada.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho a ser excluído.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do gatilho excluído.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Ação ListTriggers (Python: list_triggers)

Recupera os nomes de todos os recursos de acionador nessa conta da AWS ou os recursos com a tag especificada. Essa operação permite que você veja quais recursos estão disponíveis em sua conta e seus nomes.

Essa operação aceita o campo `Tags` opcional, que pode ser usado como um filtro na resposta, para que recursos com tags possam ser recuperados como um grupo. Se você optar por usar a filtragem por tags, apenas os recursos com a tag serão recuperados.

Solicitação

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma solicitação de continuação.

- `DependentJobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do trabalho para o qual recuperar gatilhos. O gatilho que pode iniciar esse trabalho é retornado. Se esse gatilho não existir, todos os gatilhos serão retornados.

- `MaxResults`: número (inteiro) não inferior a 1 nem superior a 200.

  O tamanho máximo de uma lista a ser retornada.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  Especifica apenas o retorno desses recursos com tags.

Resposta

- `TriggerNames` – Uma matriz de strings UTF-8.

  Os nomes de todos os gatilhos na conta ou os gatilhos com as tags especificadas.

- `NextToken` – String UTF-8.

  Um token de continuação, se a lista retornada não contiver a métrica mais recente disponível.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação BatchGetTriggers (Python: batch_get_triggers)

Retorna uma lista de metadados de recurso para uma lista de nomes de gatilho. Depois de chamar a operação `ListTriggers`, você pode chamar essa operação para acessar os dados aos quais você recebeu permissões. Essa operação oferece suporte a todas as permissões do IAM, incluindo condições de permissão que usam tags.

Solicitação

- `TriggerNames`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de nomes de gatilho, que podem ser os nomes retornados pela operação `ListTriggers`.

Resposta

- `Triggers` – Uma matriz de objetos Trigger.

  Uma lista das definições de gatilho.
- `TriggersNotFound` – Uma matriz de strings UTF-8.

  Uma lista de nomes de gatilhos não encontrados.

Erros

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Sessões API interativas

A API de sessões interativas descreve a AWS Glue API relacionada ao uso de sessões AWS Glue interativas para criar e testar scripts de extração, transformação e carregamento (ETL) para integração de dados.

Tipos de dados

- Estrutura Session
- SessionCommand estrutura
- Estrutura Statement
- StatementOutput estrutura
- StatementOutputData estrutura
- ConnectionsList estrutura

Estrutura Session

O período em que um ambiente do runtime remoto do Spark está sendo executado.

Campos

- `Id` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da sessão.
- `CreatedOn` – Timestamp.

  A data e a hora em que a sessão foi criada.
- `Status` – String UTF-8 (valores válidos: `PROVISIONING` | `READY` | `FAILED` | `TIMEOUT` | `STOPPING` | `STOPPED`).

  O status da sessão.
- `ErrorMessage` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A mensagem de erro exibida durante a sessão.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A descrição da sessão.

- `Role` – String UTF-8, superior a 20 e inferior a 2048 bytes de comprimento, correspondente a Custom string pattern #26.

  O nome ou nome de recurso da Amazon (ARN) da função do IAM associada à sessão.

- `Command` – Um objeto SessionCommand.

  O comando Object.see. SessionCommand

- `DefaultArguments` – Uma matriz de mapas de pares de valores-chave, não mais de 75 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes, correspondente a Custom string pattern #27.

  Cada valor é uma string UTF-8, com comprimento não superior a 4096 bytes, correspondente a URI address multi-line string pattern.

  Um array de mapa dos pares de valor-chave. O máximo é de 75 pares.

- `Connections` – Um objeto ConnectionsList.

  O número de conexões usadas para a sessão.

- `Progress` – Número (duplo).

  O andamento da execução do código da sessão.

- `MaxCapacity` – Número (duplo).

  O número de unidades de processamento de AWS Glue dados (DPUs) que podem ser alocadas quando o trabalho é executado. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória.

- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da SecurityConfiguration estrutura a ser usada com a sessão.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

A AWS Glue versão determina as versões do Apache Spark e do Python compatíveis. AWS Glue O GlueVersion deve ser maior que 2,0.

- `DataAccessId` - String UTF-8, não menos do que 1 ou superior a 36 bytes de comprimento.

  O ID de acesso a dados da sessão.

- `PartitionId` - String UTF-8, não menos do que 1 ou superior a 36 bytes de comprimento.

  O ID da partição da sessão.

- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de determinado `WorkerType` para usar na sessão.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido que é alocado quando uma sessão é executada. Aceita um valor de `G.1X`, `G.2X`, `G.4X` ou `G.8X` para sessões do Spark. Aceita o valor `Z.2X` para sessões do Ray.

- `CompletedOn` – Timestamp.

  A data e a hora em que a execução desse trabalho foi concluída.

- `ExecutionTime` – Número (duplo).

  O tempo total durante o qual a sessão foi executada.

- `DPUSeconds` – Número (duplo).

  As DPUs consumidas pela sessão (fórmula: ExecutionTime * MaxCapacity).

- `IdleTimeout` – Número (íntegro).

  O número de minutos de natividade até esgotar o tempo limite da sessão.

- `ProfileName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de um perfil de AWS Glue uso associado à sessão.

SessionCommand estrutura

O `SessionCommand` que executa esse trabalho.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Especifica o nome do SessionCommand. Pode ser “glueetl” ou “glueestreaming”.
- `PythonVersion` – String UTF-8 correspondente a Custom string pattern #21.

  Especifica a versão usada do Python. A versão do Python indica a versão compatível com trabalhos do tipo Spark.

Estrutura Statement

A declaração ou solicitação para que uma ação específica ocorra em uma sessão.

Campos

- `Id` – Número (íntegro).

  O ID da instrução.
- `Code` – String UTF-8.

  O código de execução da instrução.
- `State` – String UTF-8 (valores válidos: `WAITING` | `RUNNING` | `AVAILABLE` | `CANCELLING` | `CANCELLED` | `ERROR`).

  O estado enquanto a solicitação é acionada.
- `Output` – Um objeto StatementOutput.

  A saída em JSON.
- `Progress` – Número (duplo).

  O progresso da execução do código.
- `StartedOn` – Número (extenso).

  A hora e a data unix em que a definição de trabalho foi iniciada.
- `CompletedOn` – Número (extenso).

  A hora e a data unix em que a definição de trabalho foi concluída.

StatementOutput estrutura

O resultado da execução de código no formato JSON.

Campos

- `Data` – Um objeto StatementOutputData.

  O resultado da execução de código.
- `ExecutionCount` – Número (íntegro).

  A contagem de execução da saída.
- `Status` – String UTF-8 (valores válidos: `WAITING` | `RUNNING` | `AVAILABLE` | `CANCELLING` | `CANCELLED` | `ERROR`).

  O status da saída do código de execução.
- `ErrorName` – String UTF-8.

  O nome do erro na saída.
- `ErrorValue` – String UTF-8.

  O valor de erro da saída.
- `Traceback` – Uma matriz de strings UTF-8.

  O traceback da saída.

StatementOutputData estrutura

O resultado da execução de código no formato JSON.

Campos

- `TextPlain` – String UTF-8.

  O resultado da execução do código no formato de texto.

ConnectionsList estrutura

Especifica as conexões usadas por um trabalho.

Campos

- `Connections` – Uma matriz de strings UTF-8.

  Uma lista das conexões usadas pelo trabalho.

Operações

- CreateSession ação (Python: create_session)
- StopSession ação (Python: stop_session)
- DeleteSession ação (Python: delete_session)
- GetSession ação (Python: get_session)
- ListSessions ação (Python: list_sessions)
- RunStatement ação (Python: run_statement)
- CancelStatement ação (Python: cancel_statement)
- GetStatement ação (Python: get_statement)
- ListStatements ação (Python: list_statements)

CreateSession ação (Python: create_session)

Cria uma nova sessão..

Solicitação

Solicitação para criar uma nova sessão.

- `Id` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da solicitação de sessão.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A descrição da sessão.
- `Role` – Obrigatório: string UTF-8, não inferior a 20 ou superior a 2048 bytes de comprimento, correspondendo a Custom string pattern #26.

O ARN da função do &IAM

- `Command` – Obrigatório: um objeto SessionCommand.

  O `SessionCommand` que executa esse trabalho.

- `Timeout` – Número (inteiro), pelo menos 1.

  O número de minutos até esgotar o tempo limite da sessão. O padrão para trabalhos de ETL do Spark é 48 horas (2.880 minutos), a duração máxima da sessão para esse tipo de trabalho. Consulte a documentação para outros tipos de trabalho.

- `IdleTimeout` – Número (inteiro), pelo menos 1.

  O número de minutos de natividade até esgotar o tempo limite da sessão. O padrão para trabalhos de ETL do Spark é o valor do tempo limite. Consulte a documentação para outros tipos de trabalho.

- `DefaultArguments` – Uma matriz de mapas de pares de valores-chave, não mais de 75 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes, correspondente a Custom string pattern #27.

  Cada valor é uma string UTF-8, com comprimento não superior a 4096 bytes, correspondente a URI address multi-line string pattern.

  Um array de mapa dos pares de valor-chave. O máximo é de 75 pares.

- `Connections` – Um objeto ConnectionsList.

  O número de conexões usadas para a sessão.

- `MaxCapacity` – Número (duplo).

  O número de unidades de processamento de AWS Glue dados (DPUs) que podem ser alocadas quando o trabalho é executado. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória.

- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de determinado `WorkerType` para usar na sessão.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

O tipo de operador predefinido que é alocado quando um trabalho é executado. Aceita um valor de G.1X, G.2X, G.4X ou G.8X para trabalhos do Spark. Aceita o valor Z.2X para cadernos do Ray.

- Para o tipo de operador `G.1X`, cada operador é mapeado para 1 DPU (4 vCPU, 16 GB de memória) com disco de 84 GB (aproximadamente 34 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
- Para o tipo de operador `G.2X`, cada operador é mapeado para 2 DPU (8 vCPU, 32 GB de memória) com disco de 128 GB (aproximadamente 77 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para workloads, como transformações de dados, uniões e consultas, para oferecer uma maneira escalável e econômica de executar a maioria dos trabalhos.
- Para o tipo de operador `G.4X`, cada operador é mapeado para 4 DPU (16 vCPU, 64 GB de memória) com disco de 256 GB (aproximadamente 235 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark AWS nas seguintes regiões: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Cingapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central), Europa (Frankfurt), Europa (Irlanda) e Europa (Estocolmo).
- Para o tipo de operador `G.8X`, cada operador é mapeado para 8 DPU (32 vCPU, 128 GB de memória) com disco de 512 GB (aproximadamente 487 GB livres) e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos cujas workloads contêm as transformações, agregações, uniões e consultas mais exigentes. Esse tipo de trabalhador está disponível somente para trabalhos ETL da AWS Glue versão 3.0 ou posterior do Spark, nas mesmas AWS regiões compatíveis com o tipo de `G.4X` trabalhador.
- Para o tipo de operador `Z.2X`, cada operador é mapeado para 2 M-DPU (8 vCPUs, 64 GB de memória) com 128 GB de disco (aproximadamente 120 GB livres) e fornece até 8 operadores do Ray baseados no escalador automático.

- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da SecurityConfiguration estrutura a ser usada com a sessão

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

A AWS Glue versão determina as versões do Apache Spark e do Python compatíveis. AWS Glue O GlueVersion deve ser maior que 2,0.

- `DataAccessId` - String UTF-8, não menos do que 1 ou superior a 36 bytes de comprimento.

  O ID de acesso a dados da sessão.

- `PartitionId` - String UTF-8, não menos do que 1 ou superior a 36 bytes de comprimento.

  O ID da partição da sessão.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  O mapa dos pares de valores-chave (tags) pertencentes à sessão.

- `RequestOrigin`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  A origem da solicitação.

- `ProfileName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de um perfil de AWS Glue uso associado à sessão.

Resposta

- `Session` – Um objeto Sessão.

  Retorna o objeto de sessão na resposta.

Erros

- `AccessDeniedException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

- `ValidationException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`

StopSession ação (Python: stop_session)

Interrompa a sessão.

Solicitação

- `Id` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da sessão a ser interrompida.
- `RequestOrigin`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  A origem da solicitação.

Resposta

- `Id` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Retorna o ID da sessão interrompida.

Erros

- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`
- `ConcurrentModificationException`

DeleteSession ação (Python: delete_session)

Exclui a sessão.

Solicitação

- `Id` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da sessão a ser excluído.

- `RequestOrigin`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  O nome da origem da solicitação de sessão de exclusão.

Resposta

- `Id` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Retorna o ID da sessão excluída.

Erros

- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`
- `ConcurrentModificationException`

GetSession ação (Python: get_session)

Recupera a sessão.

Solicitação

- `Id` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da sessão.
- `RequestOrigin`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  A origem da solicitação.

Resposta

- `Session` – Um objeto Sessão.

  O objeto de sessão é retornado na resposta.

Erros

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

ListSessions ação (Python: list_sessions)

Recupere uma lista de sessões.

Solicitação

- `NextToken`: string UTF-8, não superior a 400.000 bytes de comprimento.

  O token para o próximo conjunto de resultados ou nulo se não houver mais resultados.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de resultados.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

Tags pertencentes à sessão.

- `RequestOrigin`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  A origem da solicitação.

Resposta

- `Ids` – Uma matriz de strings UTF-8.

  Retorna o ID da sessão.
- `Sessions` – Uma matriz de objetos Sessão.

  Retorna o objeto de sessão.
- `NextToken`: string UTF-8, não superior a 400.000 bytes de comprimento.

  O token para o próximo conjunto de resultados ou nulo se não houver mais resultados.

Erros

- `AccessDeniedException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

RunStatement ação (Python: run_statement)

Executa a instrução.

Solicitação

- `SessionId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O ID da sessão da instrução a ser executado.

- `Code`: Obrigatório: string UTF-8, não superior a 68000 bytes de comprimento.

  O código da instrução a ser executado.

- `RequestOrigin`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  A origem da solicitação.

Resposta

- `Id` – Número (íntegro).

  Retorna o ID da instrução que foi executada.

Erros

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `ResourceNumberLimitExceededException`
- `IllegalSessionStateException`

CancelStatement ação (Python: cancel_statement)

Cancela a instrução.

Solicitação

- `SessionId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da sessão da instrução a ser cancelado.

- `Id`: obrigatório: número (inteiro).

  O ID da instrução a ser cancelado.

- `RequestOrigin`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  A origem da solicitação para cancelar a instrução.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`

GetStatement ação (Python: get_statement)

Recupera a instrução.

Solicitação

- `SessionId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da sessão da instrução.

- `Id`: obrigatório: número (inteiro).

  O ID da instrução.

- `RequestOrigin`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

A origem da solicitação.

Resposta

- `Statement` – Um objeto Statement.

  Recupera a instrução.

Erros

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`

ListStatements ação (Python: list_statements)

Lista as instruções para a sessão.

Solicitação

- `SessionId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da sessão das instruções.

- `RequestOrigin`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  A origem da solicitação para listar instruções.

- `NextToken`: string UTF-8, não superior a 400.000 bytes de comprimento.

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `Statements` – Uma matriz de objetos Statement.

  Retorna a lista de instruções.
- `NextToken`: string UTF-8, não superior a 400.000 bytes de comprimento.

  Um token de continuação, se nem todas as instruções ainda tiverem sido retornadas.

Erros

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`

API de endpoints de desenvolvimento

A API Development endpoints descreve a API do AWS Glue relacionada a testes com uso de um DevEndpoint personalizado.

Tipos de dados

- Estrutura DevEndpoint
- Estrutura DevEndpointCustomLibraries

Estrutura DevEndpoint

Um endpoint de desenvolvimento no qual um desenvolvedor pode depurar scripts de extração, transformação e carregamento (ETL) remotamente.

Campos

- `EndpointName` – String UTF-8.

O nome da `DevEndpoint`.

- `RoleArn` – String UTF-8 correspondente a AWS IAM ARN string pattern.

  O nome de recurso da Amazon (ARN) da função do IAM usada nesse `DevEndpoint`.

- `SecurityGroupIds` – Uma matriz de strings UTF-8.

  Uma lista dos identificadores de grupo de segurança usados neste `DevEndpoint`.

- `SubnetId` – String UTF-8.

  O ID de sub-rede para esse `DevEndpoint`.

- `YarnEndpointAddress` – String UTF-8.

  O endereço do endpoint YARN usado por este `DevEndpoint`.

- `PrivateAddress` – String UTF-8.

  Um endereço IP privado para acessar o `DevEndpoint` em uma VPC se o `DevEndpoint` for criado dentro de um. O campo `PrivateAddress` está presente somente quando você cria o `DevEndpoint` na VPC.

- `ZeppelinRemoteSparkInterpreterPort` – Número (íntegro).

  A porta Apache Zeppelin para o intérprete Apache Spark remoto.

- `PublicAddress` – String UTF-8.

  O endereço IP público usado por esse `DevEndpoint`. O campo `PublicAddress` está presente somente quando você cria uma nuvem privada não virtual (VPC) `DevEndpoint`.

- `Status` – String UTF-8.

  O status atual deste `DevEndpoint`.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido alocado para o endpoint de desenvolvimento. Aceita um valor de Standard (Padrão), G.1X ou G.2X.

  - Para o tipo de operador `Standard`, cada operador fornece 4 vCPU, 16 GB de memória e um disco de 50 GB e 2 executores por operador.

- Para o tipo de operador `G.1X`, cada operador é mapeado para 1 DPU (4 vCPU, 16 GB de memória, disco de 64 GB), e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos com uso intensivo de memória.
- Para o tipo de operador `G.2X`, cada operador é mapeado para 2 DPU (8 vCPU, 32 GB de memória, disco de 128 GB), e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos com uso intensivo de memória.

Problema conhecido: quando um endpoint de desenvolvimento é criado com a configuração `WorkerType` `G.2X`, os drivers do Spark para o endpoint de desenvolvimento são executados em 4 vCPUs, 16 GB de memória e um disco de 64 GB.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  A versão do Glue determina as versões do Apache Spark e do Python compatíveis com o AWS Glue. A versão do Python indica a versão compatível para executar seus scripts de ETL em endpoints de desenvolvimento.

  Para obter mais informações sobre as versões disponíveis do AWS Glue e as versões correspondentes do Spark e do Python, consulte Versão do Glue no guia do desenvolvedor.

  Os endpoints de desenvolvimento criados sem especificar uma versão do Glue usam como padrão o Glue 0.9.

  É possível especificar uma versão do suporte do Python para endpoints de desenvolvimento usando o parâmetro `Arguments` nas APIs `UpdateDevEndpoint` ou `CreateDevEndpoint`. Se nenhum argumento for fornecido, a versão usará o Python 2 como padrão.
- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de um `workerType` definido que são alocados para o endpoint de desenvolvimento.

  O número máximo de operadores que você pode definir são 299 para `G.1X` e 149 para `G.2X`.
- `NumberOfNodes` – Número (íntegro).

  O número de unidades de processamento de dados (DPUs) do AWS Glue alocadas para esse `DevEndpoint`.
- `AvailabilityZone` – String UTF-8.

A zona de disponibilidade da AWS onde esse `DevEndpoint` está localizado.

- `VpcId` – String UTF-8.

  O ID da nuvem privada virtual (VPC) usada por esse `DevEndpoint`.

- `ExtraPythonLibsS3Path` – String UTF-8.

  Os caminhos para um ou mais bibliotecas Python em um bucket do Amazon S3 que deve ser carregado no seu `DevEndpoint`. Vários valores devem ser caminhos completos separados por uma vírgula.

  > **Note**
  >
  > Você pode usar apenas bibliotecas Python puras com um `DevEndpoint`. Bibliotecas que contam com extensões C, como a biblioteca de análise de dados Python 8pandas, não são compatíveis no momento.

- `ExtraJarsS3Path` – String UTF-8.

  Os caminhos para um ou mais arquivos `.jar` do Java em um bucket do S3 que deve ser carregado no `DevEndpoint`.

  > **Note**
  >
  > Você pode usar apenas bibliotecas Java/Scala com um `DevEndpoint`.

- `FailureReason` – String UTF-8.

  O motivo da falha atual neste `DevEndpoint`.

- `LastUpdateStatus` – String UTF-8.

  O status da última atualização.

- `CreatedTimestamp` – Timestamp.

  O momento em que este DevEndpoint foi criado.

- `LastModifiedTimestamp` – Timestamp.

  O momento em que esse `DevEndpoint` foi modificado pela última vez.

- `PublicKey` – String UTF-8.

A chave pública a ser usada por esse `DevEndpoint` para autenticação. Este atributo é fornecido para compatibilidade com versões anteriores, pois o atributo recomendado para uso são chaves públicas.

- `PublicKeys` – Uma matriz de strings UTF-8, no máximo 5 strings.

  Uma lista de chaves públicas a serem usadas pelos `DevEndpoints` para autenticação. O uso desse atributo é preferido em vez de uma única chave pública, pois as chaves públicas permitem que você tenha uma chave privada diferente por cliente.

  > **Note**
  >
  > Se você tiver criado um endpoint com uma chave pública, deverá remover essa chave para poder definir uma lista de chaves públicas. Chame a operação da API `UpdateDevEndpoint` com o conteúdo da chave pública no atributo `deletePublicKeys` e a lista de novas chaves no atributo `addPublicKeys`.

- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da estrutura de `SecurityConfiguration` a ser usada com esse `DevEndpoint`.

- `Arguments`: uma matriz de mapa dos pares de chave-valor, não mais do que 100 pares.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Um mapa de argumentos usados para configurar o `DevEndpoint`.

  Os argumentos válidos são:

  - `"--enable-glue-datacatalog": ""`

  É possível especificar uma versão do suporte do Python para endpoints de desenvolvimento usando o parâmetro `Arguments` nas APIs `UpdateDevEndpoint` ou `CreateDevEndpoint`. Se nenhum argumento for fornecido, a versão usará o Python 2 como padrão.

Estrutura DevEndpointCustomLibraries

Bibliotecas personalizadas a serem carregadas em um endpoint de desenvolvimento.

Campos

- `ExtraPythonLibsS3Path` – String UTF-8.

  Os caminhos para uma ou mais bibliotecas Python em um bucket do Amazon Simple Storage Service (Amazon S3) que devem ser carregadas no `DevEndpoint`. Vários valores devem ser caminhos completos separados por uma vírgula.

  > **Note**
  >
  > Você pode usar apenas bibliotecas Python puras com um `DevEndpoint`. Bibliotecas que contam com extensões C, como a biblioteca de análise de dados Python 8pandas, não são compatíveis no momento.

- `ExtraJarsS3Path` – String UTF-8.

  Os caminhos para um ou mais arquivos `.jar` do Java em um bucket do S3 que deve ser carregado no `DevEndpoint`.

  > **Note**
  >
  > Você pode usar apenas bibliotecas Java/Scala com um `DevEndpoint`.

Operações

- Ação CreateDevEndpoint (Python: create_dev_endpoint)
- Ação UpdateDevEndpoint (Python: update_dev_endpoint)
- Ação DeleteDevEndpoint (Python: delete_dev_endpoint)
- Ação GetDevEndpoint (Python: get_dev_endpoint)
- Ação GetDevEndpoints (Python: get_dev_endpoints)
- Ação BatchGetDevEndpoints (Python: batch_get_dev_endpoints)
- Ação ListDevEndpoints (Python: list_dev_endpoints)

Ação CreateDevEndpoint (Python: create_dev_endpoint)

Cria um endpoint de desenvolvimento.

Solicitação

- `EndpointName` – Obrigatório: string UTF-8.

  O nome a ser atribuído ao novo `DevEndpoint`.

- `RoleArn` – Obrigatório: string UTF-8, correspondente a AWS IAM ARN string pattern.

  A função do IAM para o `DevEndpoint`.

- `SecurityGroupIds` – Uma matriz de strings UTF-8.

  IDs dos grupos de segurança a serem usados pelo novo `DevEndpoint`.

- `SubnetId` – String UTF-8.

  O ID de sub-rede para o novo `DevEndpoint` a ser usado.

- `PublicKey` – String UTF-8.

  A chave pública a ser usada por esse `DevEndpoint` para autenticação. Este atributo é fornecido para compatibilidade com versões anteriores, pois o atributo recomendado para uso são chaves públicas.

- `PublicKeys` – Uma matriz de strings UTF-8, no máximo 5 strings.

  Uma lista de chaves públicas a serem usadas pelos endpoints de desenvolvimento para autenticação. O uso desse atributo é preferido em vez de uma única chave pública, pois as chaves públicas permitem que você tenha uma chave privada diferente por cliente.

  > **Note**
  >
  > Se você tiver criado um endpoint com uma chave pública, deverá remover essa chave para poder definir uma lista de chaves públicas. Chame a API `UpdateDevEndpoint` com o conteúdo da chave pública no atributo `deletePublicKeys` e a lista de novas chaves no atributo `addPublicKeys`.

- `NumberOfNodes` – Número (íntegro).

  O número de unidades de processamento de dados (DPUs) do AWS Glue a serem alocadas para esse `DevEndpoint`.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

O tipo de operador predefinido alocado para o endpoint de desenvolvimento. Aceita um valor de Standard (Padrão), G.1X ou G.2X.

- Para o tipo de operador `Standard`, cada operador fornece 4 vCPU, 16 GB de memória e um disco de 50 GB e 2 executores por operador.
- Para o tipo de operador `G.1X`, cada operador é mapeado para 1 DPU (4 vCPU, 16 GB de memória, disco de 64 GB), e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos com uso intensivo de memória.
- Para o tipo de operador `G.2X`, cada operador é mapeado para 2 DPU (8 vCPU, 32 GB de memória, disco de 128 GB), e fornece 1 executor por operador. Recomendamos esse tipo de operador para trabalhos com uso intensivo de memória.

Problema conhecido: quando um endpoint de desenvolvimento é criado com a configuração `WorkerType` `G.2X`, os drivers do Spark para o endpoint de desenvolvimento são executados em 4 vCPUs, 16 GB de memória e um disco de 64 GB.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  A versão do Glue determina as versões do Apache Spark e do Python compatíveis com o AWS Glue. A versão do Python indica a versão compatível para executar seus scripts de ETL em endpoints de desenvolvimento.

  Para obter mais informações sobre as versões disponíveis do AWS Glue e as versões correspondentes do Spark e do Python, consulte Versão do Glue no guia do desenvolvedor.

  Os endpoints de desenvolvimento criados sem especificar uma versão do Glue usam como padrão o Glue 0.9.

  É possível especificar uma versão do suporte do Python para endpoints de desenvolvimento usando o parâmetro `Arguments` nas APIs `UpdateDevEndpoint` ou `CreateDevEndpoint`. Se nenhum argumento for fornecido, a versão usará o Python 2 como padrão.

- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de um `workerType` definido que são alocados para o endpoint de desenvolvimento.

  O número máximo de operadores que você pode definir são 299 para `G.1X` e 149 para `G.2X`.

- `ExtraPythonLibsS3Path` – String UTF-8.

Os caminhos para um ou mais bibliotecas Python em um bucket do Amazon S3 que deve ser carregado no seu `DevEndpoint`. Vários valores devem ser caminhos completos separados por uma vírgula.

> **Note**
>
> Você pode usar apenas bibliotecas Python puras com um `DevEndpoint`. Bibliotecas que contam com extensões C, como a biblioteca de análise de dados Python pandas, ainda não são compatíveis.

- `ExtraJarsS3Path` – String UTF-8.

  Os caminhos para um ou mais arquivos `.jar` do Java em um bucket do S3 que deve ser carregado no `DevEndpoint`.

- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da estrutura de `SecurityConfiguration` a ser usada com esse `DevEndpoint`.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags a serem usadas com esse DevEndpoint. Você pode usar tags para limitar o acesso ao DevEndpoint. Para obter mais informações sobre tags no AWS Glue, consulte Tags da AWS no AWS Glue no guia do desenvolvedor.

- `Arguments`: uma matriz de mapa dos pares de chave-valor, não mais do que 100 pares.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  Um mapa de argumentos usados para configurar o `DevEndpoint`.

Resposta

- `EndpointName` – String UTF-8.

O nome atribuído ao novo `DevEndpoint`.

- `Status` – String UTF-8.

  O status atual do novo `DevEndpoint`.

- `SecurityGroupIds` – Uma matriz de strings UTF-8.

  Os grupos de segurança atribuídos ao novo `DevEndpoint`.

- `SubnetId` – String UTF-8.

  O ID de sub-rede atribuído ao novo `DevEndpoint`.

- `RoleArn` – String UTF-8 correspondente a AWS IAM ARN string pattern.

  O nome de recurso da Amazon (ARN) da função atribuída ao novo `DevEndpoint`.

- `YarnEndpointAddress` – String UTF-8.

  O endereço do endpoint YARN usado por esse `DevEndpoint`.

- `ZeppelinRemoteSparkInterpreterPort` – Número (íntegro).

  A porta Apache Zeppelin para o intérprete Apache Spark remoto.

- `NumberOfNodes` – Número (íntegro).

  O número de unidades de processamento de dados (DPUs) do AWS Glue alocadas para esse DevEndpoint.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido alocado para o endpoint de desenvolvimento. Deve ser um valor de Standard (Padrão), G.1X ou G.2X.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  A versão do Glue determina as versões do Apache Spark e do Python compatíveis com o AWS Glue. A versão do Python indica a versão compatível para executar seus scripts de ETL em endpoints de desenvolvimento.

  Para obter mais informações sobre as versões disponíveis do AWS Glue e as versões correspondentes do Spark e do Python, consulte Versão do Glue no guia do desenvolvedor.

- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de um `workerType` definido que são alocados para o endpoint de desenvolvimento.

- `AvailabilityZone` – String UTF-8.

  A zona de disponibilidade da AWS onde esse `DevEndpoint` está localizado.

- `VpcId` – String UTF-8.

  O ID da nuvem privada virtual (VPC) usada por esse `DevEndpoint`.

- `ExtraPythonLibsS3Path` – String UTF-8.

  Os caminhos para um ou mais bibliotecas Python em um bucket do S3 que será carregado no `DevEndpoint`.

- `ExtraJarsS3Path` – String UTF-8.

  Os caminhos para um ou mais arquivos `.jar` do Java em um bucket do S3 que será carregado no `DevEndpoint`.

- `FailureReason` – String UTF-8.

  O motivo da falha atual neste `DevEndpoint`.

- `SecurityConfiguration` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da estrutura de `SecurityConfiguration` a ser usada com esse `DevEndpoint`.

- `CreatedTimestamp` – Timestamp.

  O momento em que esse `DevEndpoint` foi criado.

- `Arguments`: uma matriz de mapa dos pares de chave-valor, não mais do que 100 pares.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  O mapa de argumentos usados para configurar esse `DevEndpoint`.

  Os argumentos válidos são:

  - `"--enable-glue-datacatalog": ""`

É possível especificar uma versão do suporte do Python para endpoints de desenvolvimento usando o parâmetro `Arguments` nas APIs `UpdateDevEndpoint` ou `CreateDevEndpoint`. Se nenhum argumento for fornecido, a versão usará o Python 2 como padrão.

Erros

- `AccessDeniedException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `ResourceNumberLimitExceededException`

Ação UpdateDevEndpoint (Python: update_dev_endpoint)

Atualiza um endpoint de desenvolvimento especificado.

Solicitação

- `EndpointName` – Obrigatório: string UTF-8.

  O nome do `DevEndpoint` a ser atualizado.
- `PublicKey` – String UTF-8.

  A chave pública a ser usada pelo `DevEndpoint`.
- `AddPublicKeys` – Uma matriz de strings UTF-8, no máximo 5 strings.

  A lista de chaves públicas a serem usadas pelo `DevEndpoint`.
- `DeletePublicKeys` – Uma matriz de strings UTF-8, no máximo 5 strings.

  A lista de chaves públicas a serem excluídas do `DevEndpoint`.
- `CustomLibraries` – Um objeto DevEndpointCustomLibraries.

Bibliotecas personalizadas Python ou Java a serem carregadas no `DevEndpoint`.

- `UpdateEtlLibraries` – Booleano.

  `True`, se a lista de bibliotecas personalizadas a serem carregadas no endpoint de desenvolvimento precisarem ser atualizadas, ou `False`, caso contrário.

- `DeleteArguments` – Uma matriz de strings UTF-8.

  A lista de chaves de argumentos a serem excluídos do mapa de argumentos usado para configurar o `DevEndpoint`.

- `AddArguments`: uma matriz de mapa dos pares de chave-valor, não mais do que 100 pares.

  Cada chave é uma sequência de caracteres UTF-8.

  Cada valor é uma sequência de caracteres UTF-8.

  O mapa de argumentos para adicionar o mapa de argumentos usado para configurar o `DevEndpoint`.

  Os argumentos válidos são:

  - `"--enable-glue-datacatalog": ""`

  É possível especificar uma versão do suporte do Python para endpoints de desenvolvimento usando o parâmetro `Arguments` nas APIs `UpdateDevEndpoint` ou `CreateDevEndpoint`. Se nenhum argumento for fornecido, a versão usará o Python 2 como padrão.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`

Ação DeleteDevEndpoint (Python: delete_dev_endpoint)

Exclui um endpoint de desenvolvimento especificado.

Solicitação

- `EndpointName` – Obrigatório: string UTF-8.

  O nome da `DevEndpoint`.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Ação GetDevEndpoint (Python: get_dev_endpoint)

Recupera informações sobre um endpoint de desenvolvimento especificado.

> **Note**
>
> Quando você cria um endpoint de desenvolvimento em uma nuvem privada virtual (VPC), o AWS Glue retorna apenas um endereço IP privado e o campo do endereço IP público campo não é preenchido. Quando você cria um endpoint de desenvolvimento não seja da VPC, o AWS Glue retorna somente um endereço IP público.

Solicitação

- `EndpointName` – Obrigatório: string UTF-8.

  Nome do `DevEndpoint` para o qual recuperar informações.

Resposta

- `DevEndpoint` – Um objeto DevEndpoint.

  Uma definição de `DevEndpoint`.

Erros

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Ação GetDevEndpoints (Python: get_dev_endpoints)

Recupera todos os endpoints de desenvolvimento nessa conta da AWS.

> **Note**
>
> Quando você cria um endpoint de desenvolvimento em uma nuvem privada virtual (VPC), o AWS Glue retorna apenas um endereço IP privado, e o campo do endereço IP público campo não é preenchido. Quando você cria um endpoint de desenvolvimento não seja da VPC, o AWS Glue retorna somente um endereço IP público.

Solicitação

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo das informações a serem retornadas.
- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `DevEndpoints` – Uma matriz de objetos DevEndpoint.

Uma lista de definições de `DevEndpoint`.

- `NextToken` – String UTF-8.

  Um token de continuação, caso nem todas as definições de `DevEndpoint` tenham sido retornadas ainda.

Erros

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Ação BatchGetDevEndpoints (Python: batch_get_dev_endpoints)

Retorna uma lista de metadados de recurso para uma lista de nomes de endpoints de desenvolvimento. Depois de chamar a operação `ListDevEndpoints`, você pode chamar essa operação para acessar os dados aos quais você recebeu permissões. Essa operação oferece suporte a todas as permissões do IAM, incluindo condições de permissão que usam tags.

Solicitação

- `customerAccountId` – String UTF-8.

  O ID da conta da AWS.

- `DevEndpointNames` – Obrigatório: uma matriz de strings UTF-8, no mínimo 1 ou mais de 25 strings.

  Uma lista de nomes de `DevEndpoint`, que podem ser os nomes retornados da operação `ListDevEndpoint`.

Resposta

- `DevEndpoints` – Uma matriz de objetos DevEndpoint.

  Uma lista de definições de `DevEndpoint`.

- `DevEndpointsNotFound`: uma matriz de strings UTF-8, não menos do que 1 ou superior a 25 strings.

  Uma lista de `DevEndpoints` não encontrados.

Erros

- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Ação ListDevEndpoints (Python: list_dev_endpoints)

Recupera os nomes de todos os recursos do `DevEndpoint` nessa conta da AWS ou os recursos com a tag especificada. Essa operação permite que você veja quais recursos estão disponíveis em sua conta e seus nomes.

Essa operação aceita o campo `Tags` opcional, que pode ser usado como um filtro na resposta, para que recursos com tags possam ser recuperados como um grupo. Se você optar por usar a filtragem por tags, apenas os recursos com a tag serão recuperados.

Solicitação

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma solicitação de continuação.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo de uma lista a ser retornada.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  Especifica apenas o retorno desses recursos com tags.

Resposta

- `DevEndpointNames` – Uma matriz de strings UTF-8.

  Os nomes de cada `DevEndpoint` na conta de cada `DevEndpoint` com as tags especificadas.
- `NextToken` – String UTF-8.

  Um token de continuação, se a lista retornada não contiver a métrica mais recente disponível.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Registro de esquemas

A API de registro de esquemas descreve os tipos de dados e a API relacionados ao trabalho com esquemas em. AWS Glue

Tipos de dados

- RegistryId estrutura
- RegistryListItem estrutura
- MetadataInfo estrutura
- OtherMetadataValueListItem estrutura
- SchemaListItem estrutura
- SchemaVersionListItem estrutura
- MetadataKeyValuePair estrutura
- SchemaVersionErrorItem estrutura
- ErrorDetails estrutura
- SchemaVersionNumber estrutura
- SchemaId estrutura

RegistryId estrutura

Uma estrutura de wrapper que pode conter o nome de registro e o nome do recurso da Amazon (ARN).

Campos

- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  Nome do registro. Usado apenas para pesquisa. Um de `RegistryArn` ou `RegistryName` tem de ser fornecido.
- `RegistryArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  ARN do registro a ser atualizado. Um de `RegistryArn` ou `RegistryName` tem de ser fornecido.

RegistryListItem estrutura

Uma estrutura que contém os detalhes de um registro.

Campos

- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro.
- `RegistryArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do registro.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do registro.
- `Status`: string UTF-8 (valores válidos: `AVAILABLE` | `DELETING`).

  O status do registro.
- `CreatedTime` – String UTF-8.

A data em que o registro foi criado.

- `UpdatedTime` – String UTF-8.

  A data em que o registro foi atualizado.

MetadataInfo estrutura

Uma estrutura que contém informações de metadados para uma versão do esquema.

Campos

- `MetadataValue`: string UTF-8, não menos do que 1 ou superior a 256 bytes de comprimento, correspondente a Custom string pattern #33.

  O valor correspondente da chave de metadados.

- `CreatedTime` – String UTF-8.

  A hora em que a entrada foi criada.

- `OtherMetadataValueList` – Uma matriz de objetos OtherMetadataValueListItem.

  Outros metadados pertencentes à mesma chave de metadados.

OtherMetadataValueListItem estrutura

Uma estrutura que contém outros metadados para uma versão do esquema pertencente à mesma chave de metadados.

Campos

- `MetadataValue`: string UTF-8, não menos do que 1 ou superior a 256 bytes de comprimento, correspondente a Custom string pattern #33.

  O valor correspondente da chave de metadados para os outros metadados pertencentes à mesma chave de metadados.

- `CreatedTime` – String UTF-8.

  A hora em que a entrada foi criada.

SchemaListItem estrutura

Um objeto que contém detalhes mínimos para um esquema.

Campos

- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro em que o esquema reside.

- `SchemaName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do esquema.

- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome do recurso da Amazon (ARN) do esquema.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição para o esquema.

- `SchemaStatus` – String UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `DELETING`).

  O status do esquema.

- `CreatedTime` – String UTF-8.

  A data e a hora em que o esquema foi criado.

- `UpdatedTime` – String UTF-8.

  A data e a hora em que o esquema foi atualizado.

SchemaVersionListItem estrutura

Um objeto que contém os detalhes sobre uma versão do esquema.

Campos

- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do esquema.
- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O identificador exclusivo da versão do esquema.
- `VersionNumber`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  O número da versão do esquema.
- `Status` – String UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  O status da versão do esquema.
- `CreatedTime` – String UTF-8.

  A data e hora em que a versão do esquema foi criada.

MetadataKeyValuePair estrutura

Uma estrutura que contém um par de chave-valor para metadados.

Campos

- `MetadataKey`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #33.

  Uma chave de metadados.
- `MetadataValue`: string UTF-8, não menos do que 1 ou superior a 256 bytes de comprimento, correspondente a Custom string pattern #33.

  O valor correspondente de uma chave de metadados.

SchemaVersionErrorItem estrutura

Um objeto que contém os detalhes de erro de uma operação em uma versão do esquema.

Campos

- `VersionNumber`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  O número da versão do esquema.
- `ErrorDetails` – Um objeto ErrorDetails.

  Os detalhes do erro para a versão do esquema.

ErrorDetails estrutura

Um objeto que contém detalhes do erro.

Campos

- `ErrorCode` – String UTF-8.

  O código de erro para um erro.
- `ErrorMessage` – String UTF-8.

  A mensagem de erro para um erro.

SchemaVersionNumber estrutura

Uma estrutura que contém as informações da versão do esquema.

Campos

- `LatestVersion` – Booleano.

  A versão mais recente disponível para o esquema.
- `VersionNumber`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  O número da versão do esquema.

SchemaId estrutura

O ID exclusivo do esquema no registro do AWS Glue esquema.

Campos

- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do esquema. Um de `SchemaArn` ou `SchemaName` tem de ser fornecido.
- `SchemaName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do esquema. Um de `SchemaArn` ou `SchemaName` tem de ser fornecido.
- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro do esquema que contém o esquema.

Operações

- CreateRegistry ação (Python: create_registry)
- CreateSchema ação (Python: create_schema)
- GetSchema ação (Python: get_schema)
- ListSchemaVersions ação (Python: list_schema_versions)
- GetSchemaVersion ação (Python: get_schema_version)
- GetSchemaVersionsDiff ação (Python: get_schema_versions_diff)
- ListRegistries ação (Python: list_registries)
- ListSchemas ação (Python: list_schemas)
- RegisterSchemaVersion ação (Python: register_schema_version)
- UpdateSchema ação (Python: update_schema)
- CheckSchemaVersionValidity ação (Python: check_schema_version_validity)
- UpdateRegistry ação (Python: update_registry)
- GetSchemaByDefinition ação (Python: get_schema_by_definition)
- GetRegistry ação (Python: get_registry)
- PutSchemaVersionMetadata ação (Python: put_schema_version_metadata)
- QuerySchemaVersionMetadata ação (Python: query_schema_version_metadata)

- RemoveSchemaVersionMetadata ação (Python: remove_schema_version_metadata)
- DeleteRegistry ação (Python: delete_registry)
- DeleteSchema ação (Python: delete_schema)
- DeleteSchemaVersions ação (Python: delete_schema_versions)

CreateRegistry ação (Python: create_registry)

Cria um novo registro que pode ser usado para armazenar uma coleção de esquemas.

Solicitação

- `RegistryName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Custom string pattern #18.

  Nome do esquema a ser criado com comprimento máximo de 255 e que só pode conter letras, números, hífen, sublinhado, sinal de dólar ou cerquilha. Sem espaço em branco
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do registro. Se a descrição não for fornecida, não haverá nenhum valor padrão para esse campo.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  AWS tags que contêm um par de valores-chave e podem ser pesquisadas por console, linha de comando ou API.

Resposta

- `RegistryArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome do recurso da Amazon (ARN) do registro recém-criado.
- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

O nome do registro.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do registro.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags do registro.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`
- `InternalServiceException`

CreateSchema ação (Python: create_schema)

Cria um novo conjunto de esquemas e registra a definição do esquema. Retorna um erro se o conjunto de esquemas já existir sem realmente registrar a versão.

Quando o conjunto de esquemas é criado, um ponto de verificação de versão é definido para a primeira versão. O modo de compatibilidade “DISABLED” (Desabilitado) restringe a adição de versões adicionais do esquema após a primeira versão. Para todos os outros modos de compatibilidade, a validação das configurações de compatibilidade será aplicada somente a partir da segunda versão em diante, quando a API do `RegisterSchemaVersion` for usada.

Quando esta API é chamada sem um `RegistryId`, isso cria uma entrada para um “default-registry” (registro padrão) nas tabelas de banco de dados do registro, se ela ainda não estiver presente.

Solicitação

- `RegistryId` – Um objeto RegistryId.

  Esta é uma forma de wrapper para conter os campos de identidade do registro. Se essa propriedade não for definida, o registro padrão será usado. O formato do ARN para a mesma propriedade será: `arn:aws:glue:us-east-2:<customer id>:registry/default-registry:random-5-letter-id`.

- `SchemaName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Custom string pattern #18.

  Nome do esquema a ser criado com comprimento máximo de 255, e só pode conter letras, números, hífen, sublinhado, sinal de dólar ou marca de hash. Sem espaço em branco

- `DataFormat` – Obrigatório: string UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  O formato de dados da definição do esquema. No momento, há suporte a `AVRO`, `JSON` e `PROTOBUF`.

- `Compatibility` – String UTF-8 (valores válidos: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  O modo de compatibilidade do esquema. Os valores possíveis são:

  - NONE (Nenhum): nenhum modo de compatibilidade se aplica. Você pode usar essa opção em cenários de desenvolvimento ou se você não souber o modo de compatibilidade que deseja aplicar aos esquemas. Qualquer nova versão adicionada será aceita sem ser submetida a uma verificação de compatibilidade.
  - DISABLED (Desabilitado): esta opção de compatibilidade impede o versionamento de um esquema específico. Você pode usar essa opção para evitar versionamentos futuros de um esquema.
  - BACKWARD (Anterior): esta opção de compatibilidade é recomendada, pois permite que os receptores de dados leiam a versão atual e uma anterior do esquema. Isso significa que, por exemplo, uma nova versão do esquema não pode descartar campos de dados ou alterar o tipo desses campos, portanto, eles não podem ser lidos pelos leitores usando a versão anterior.
  - BACKWARD_ALL (Todas as anteriores): esta opção de compatibilidade é recomendada, pois permite que os receptores de dados leiam a versão atual e todas as anteriores do esquema. Você pode usar essa opção quando precisar excluir campos ou adicionar campos opcionais e verificar a compatibilidade com todas as versões anteriores do esquema.

  - FORWARD (Próxima): esta opção de compatibilidade permite que os receptores de dados leiam a versão atual e uma próxima do esquema, mas não necessariamente versões posteriores. Você pode usar essa opção quando precisar adicionar campos ou excluir campos opcionais, mas apenas verificar a compatibilidade com a última versão do esquema.
  - FORWARD_ALL (Todas as próximas): esta opção de compatibilidade permite que os receptores de dados leiam dados gravados pelos produtores de qualquer novo esquema registrado. Você pode usar essa opção quando precisar adicionar campos ou excluir campos opcionais e verificar a compatibilidade com todas as versões anteriores do esquema.
  - FULL (Completo): esta opção de compatibilidade permite que os receptores de dados leiam dados gravados por produtores usando a versão anterior ou seguinte do esquema, mas não necessariamente versões anteriores ou posteriores. Você pode usar essa opção quando precisar adicionar ou remover campos opcionais, mas apenas verificar a compatibilidade com a última versão do esquema.
  - FULL_ALL (Completo total): esta opção de compatibilidade permite que os receptores de dados leiam dados gravados por produtores usando todas as versões de esquema anteriores. Você pode usar essa opção quando precisar adicionar ou remover campos opcionais e verificar a compatibilidade com todas as versões anteriores do esquema.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição opcional do esquema. Se a descrição não for fornecida, não haverá nenhum valor padrão automático para esse campo.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  AWS tags que contêm um par de valores-chave e podem ser pesquisadas por console, linha de comando ou API. Se especificado, segue o AWS tags-on-create padrão.
- `SchemaDefinition`: string UTF-8, não menos do que 1 ou superior a 170.000 bytes de comprimento, correspondente a Custom string pattern #32.

  A definição do esquema usando a configuração `DataFormat` para `SchemaName`.

Resposta

- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro.
- `RegistryArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do registro.
- `SchemaName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do esquema.
- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do esquema.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do esquema, se especificado quando criado.
- `DataFormat` – String UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  O formato de dados da definição do esquema. No momento, há suporte a `AVRO`, `JSON` e `PROTOBUF`.
- `Compatibility` – String UTF-8 (valores válidos: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  O modo de compatibilidade do esquema.
- `SchemaCheckpoint`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  O número da versão do ponto de verificação (a última vez que o modo de compatibilidade foi alterado).
- `LatestSchemaVersion`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  A versão mais recente do esquema associado à definição do esquema retornado.
- `NextSchemaVersion`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

A próxima versão do esquema associado à definição do esquema retornado.

- `SchemaStatus` – String UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `DELETING`).

  O status do esquema.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags do esquema.

- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O identificador exclusivo da primeira versão do esquema.

- `SchemaVersionStatus` – String UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  O status da primeira versão do esquema criada.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`
- `InternalServiceException`

GetSchema ação (Python: get_schema)

Descreve o esquema especificado em detalhes.

Solicitação

- `SchemaId` – Obrigatório: um objeto SchemaId.

  É uma estrutura de wrapper para conter campos de identidade de esquema. A estrutura contém:

  - SchemaId$SchemaArn: O nome de recurso da Amazon (ARN) do esquema. `SchemaArn` ou `SchemaName` e `RegistryName` devem ser fornecidos.
  - SchemaId$SchemaName: O nome do esquema. `SchemaArn` ou `SchemaName` e `RegistryName` devem ser fornecidos.

Resposta

- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro.
- `RegistryArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do registro.
- `SchemaName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do esquema.
- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do esquema.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do esquema, se especificado quando criado
- `DataFormat` – String UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  O formato de dados da definição do esquema. No momento, há suporte a `AVRO`, `JSON` e `PROTOBUF`.

- `Compatibility` – String UTF-8 (valores válidos: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  O modo de compatibilidade do esquema.

- `SchemaCheckpoint`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  O número da versão do ponto de verificação (a última vez que o modo de compatibilidade foi alterado).

- `LatestSchemaVersion`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  A versão mais recente do esquema associado à definição do esquema retornado.

- `NextSchemaVersion`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  A próxima versão do esquema associado à definição do esquema retornado.

- `SchemaStatus` – String UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `DELETING`).

  O status do esquema.

- `CreatedTime` – String UTF-8.

  A data e a hora em que o esquema foi criado.

- `UpdatedTime` – String UTF-8.

  A data e a hora em que o esquema foi atualizado.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

ListSchemaVersions ação (Python: list_schema_versions)

Retorna uma lista de versões de esquema que você criou, com informações mínimas. As versões do esquema no status Deleted (Excluído) não serão incluídas nos resultados. Resultados vazios serão retornados, se não houver versões de esquema disponíveis.

Solicitação

- `SchemaId` – Obrigatório: um objeto SchemaId.

  É uma estrutura de wrapper para conter campos de identidade de esquema. A estrutura contém:

  - SchemaId$SchemaArn: O nome de recurso da Amazon (ARN) do esquema. `SchemaArn` ou `SchemaName` e `RegistryName` devem ser fornecidos.
  - SchemaId$SchemaName: O nome do esquema. `SchemaArn` ou `SchemaName` e `RegistryName` devem ser fornecidos.
- `MaxResults`: número (inteiro), não menos do que 1 ou superior a 100.

  O número máximo de resultados exigidos por página. Se o valor não for fornecido, esse valor será definido por padrão como 25 por página.
- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `Schemas` – Uma matriz de objetos SchemaVersionListItem.

  Uma matriz de objetos `SchemaVersionList` contendo detalhes de cada versão do esquema.
- `NextToken` – String UTF-8.

  Um token de continuação para paginação da lista de tokens retornada, retornado se o segmento atual da lista não for o último.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetSchemaVersion ação (Python: get_schema_version)

Obtém o esquema especificado por seu ID exclusivo atribuído quando uma versão do esquema é criada ou registrada. As versões do esquema no status Deleted (Excluído) não serão incluídas nos resultados.

Solicitação

- `SchemaId` – Um objeto SchemaId.

  É uma estrutura de wrapper para conter campos de identidade de esquema. A estrutura contém:

  - SchemaId$SchemaArn: O nome de recurso da Amazon (ARN) do esquema. `SchemaArn` ou `SchemaName` e `RegistryName` devem ser fornecidos.
  - SchemaId$SchemaName: O nome do esquema. `SchemaArn` ou `SchemaName` e `RegistryName` devem ser fornecidos.
- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O `SchemaVersionId` da versão do esquema. Esse campo é obrigatório para buscar pelo ID do esquema. Esse ou o wrapper `SchemaId` tem que ser fornecido.
- `SchemaVersionNumber` – Um objeto SchemaVersionNumber.

  O número da versão do esquema.

Resposta

- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O `SchemaVersionId` da versão do esquema.
- `SchemaDefinition`: string UTF-8, não menos do que 1 ou superior a 170.000 bytes de comprimento, correspondente a Custom string pattern #32.

  A definição do esquema para o ID do esquema.
- `DataFormat` – String UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  O formato de dados da definição do esquema. No momento, há suporte a `AVRO`, `JSON` e `PROTOBUF`.

- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do esquema.

- `VersionNumber`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  O número da versão do esquema.

- `Status` – String UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  O status da versão do esquema.

- `CreatedTime` – String UTF-8.

  A data e hora em que a versão do esquema foi criada.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetSchemaVersionsDiff ação (Python: get_schema_versions_diff)

Obtém a diferença de versão do esquema no tipo de diferença especificado entre duas versões de esquema armazenadas no registro de esquema.

Essa API permite comparar duas versões de esquema entre duas definições sob o mesmo esquema.

Solicitação

- `SchemaId` – Obrigatório: um objeto SchemaId.

  É uma estrutura de wrapper para conter campos de identidade de esquema. A estrutura contém:

  - SchemaId$SchemaArn: O nome de recurso da Amazon (ARN) do esquema. Um de `SchemaArn` ou `SchemaName` tem de ser fornecido.
  - SchemaId$SchemaName: O nome do esquema. Um de `SchemaArn` ou `SchemaName` tem de ser fornecido.

- `FirstSchemaVersionNumber` – Obrigatório: um objeto SchemaVersionNumber.

  A primeira das duas versões de esquema a serem comparadas.

- `SecondSchemaVersionNumber` – Obrigatório: um objeto SchemaVersionNumber.

  A segunda das duas versões de esquema a serem comparadas.

- `SchemaDiffType`: obrigatório: string UTF-8 (valores válidos: `SYNTAX_DIFF`).

  Refere-se a `SYNTAX_DIFF`, que é o tipo de comparação atualmente suportado.

Resposta

- `Diff`: string UTF-8, não menos do que 1 ou superior a 340.000 bytes de comprimento, correspondente a Custom string pattern #32.

  A diferença entre esquemas como uma string no JsonPatch formato.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`

ListRegistries ação (Python: list_registries)

Retorna uma lista de registros que você criou, com informações mínimas de registro. Registros no status `Deleting` não serão incluídos nos resultados. Resultados vazios serão retornados, se não houver registros disponíveis.

Solicitação

- `MaxResults`: número (inteiro), não menos do que 1 ou superior a 100.

  O número máximo de resultados exigidos por página. Se o valor não for fornecido, esse valor será definido por padrão como 25 por página.

- `NextToken` – String UTF-8.

Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `Registries` – Uma matriz de objetos RegistryListItem.

  Uma matriz de objetos `RegistryDetailedListItem` contendo detalhes mínimos de cada registro.
- `NextToken` – String UTF-8.

  Um token de continuação para paginação da lista de tokens retornada, retornado se o segmento atual da lista não for o último.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

ListSchemas ação (Python: list_schemas)

Retorna uma lista de esquemas com detalhes mínimos. Esquemas no status Deleting (Exclusão) não serão incluídos nos resultados. Resultados vazios serão retornados, se não houver esquemas disponíveis.

Quando o `RegistryId` não for fornecido, todos os esquemas nos registros serão parte da resposta da API.

Solicitação

- `RegistryId` – Um objeto RegistryId.

  Uma estrutura de wrapper que pode conter o nome de registro e o nome do recurso da Amazon (ARN).
- `MaxResults`: número (inteiro), não menos do que 1 ou superior a 100.

O número máximo de resultados exigidos por página. Se o valor não for fornecido, esse valor será definido por padrão como 25 por página.

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `Schemas` – Uma matriz de objetos SchemaListItem.

  Uma matriz de objetos `SchemaListItem` contendo detalhes de cada esquema.

- `NextToken` – String UTF-8.

  Um token de continuação para paginação da lista de tokens retornada, retornado se o segmento atual da lista não for o último.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

RegisterSchemaVersion ação (Python: register_schema_version)

Adiciona uma nova versão ao esquema existente. Retorna um erro se a nova versão do esquema não atender aos requisitos de compatibilidade do conjunto de esquemas. Essa API não criará um novo conjunto de esquemas e retornará um erro 404 se o conjunto de esquemas ainda não estiver presente no registro de esquema.

Se essa for a primeira definição de esquema a ser registrada no registro de esquema, essa API armazenará a versão do esquema e retornará imediatamente. Caso contrário, essa chamada tem o potencial de ser executada por mais tempo do que outras operações graças aos modos de compatibilidade. Você pode chamar a API `GetSchemaVersion` com o `SchemaVersionId` para conferir os modos de compatibilidade.

Se a mesma definição de esquema já estiver armazenada no registro de esquema como uma versão, o ID do esquema existente será retornado ao autor da chamada.

Solicitação

- `SchemaId` – Obrigatório: um objeto SchemaId.

  É uma estrutura de wrapper para conter campos de identidade de esquema. A estrutura contém:

  - SchemaId$SchemaArn: O nome de recurso da Amazon (ARN) do esquema. `SchemaArn` ou `SchemaName` e `RegistryName` devem ser fornecidos.
  - SchemaId$SchemaName: O nome do esquema. `SchemaArn` ou `SchemaName` e `RegistryName` devem ser fornecidos.
- `SchemaDefinition`: obrigatório: string UTF-8, não menos do que 1 ou superior a 170.000 bytes de comprimento, correspondente a Custom string pattern #32.

  A definição do esquema usando a configuração `DataFormat` para o `SchemaName`.

Resposta

- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O ID exclusivo que representa a versão desse esquema.
- `VersionNumber`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  A versão desse esquema (apenas para fluxo de sincronização, caso esta seja a primeira versão).
- `Status` – String UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  O status da versão do esquema.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

- `InternalServiceException`

UpdateSchema ação (Python: update_schema)

Atualiza a descrição, a configuração de compatibilidade ou o ponto de verificação de versão de um conjunto de esquemas.

Para atualizar a configuração de compatibilidade, a chamada não validará a compatibilidade para todo o conjunto de versões de esquema com a nova configuração de compatibilidade. Se o valor para `Compatibility` for fornecido, o `VersionNumber` (um ponto de verificação) também será necessário. A API validará o número de versão do ponto de verificação para fins de consistência.

Se o valor para `VersionNumber` (ponto de verificação) for fornecido, `Compatibility` será opcional e isso poderá ser usado para definir/redefinir um ponto de verificação para o esquema.

Essa atualização ocorrerá somente se o esquema estiver no estado AVAILABLE (Disponível).

Solicitação

- `SchemaId` – Obrigatório: um objeto SchemaId.

  É uma estrutura de wrapper para conter campos de identidade de esquema. A estrutura contém:

  - SchemaId$SchemaArn: O nome de recurso da Amazon (ARN) do esquema. Um de `SchemaArn` ou `SchemaName` tem de ser fornecido.
  - SchemaId$SchemaName: O nome do esquema. Um de `SchemaArn` ou `SchemaName` tem de ser fornecido.
- `SchemaVersionNumber` – Um objeto SchemaVersionNumber.

  Número de versão necessário para criar o ponto de verificação. Um de `VersionNumber` ou `Compatibility` tem de ser fornecido.
- `Compatibility` – String UTF-8 (valores válidos: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  A nova configuração de compatibilidade do esquema.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A nova descrição do esquema.

Resposta

- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do esquema.

- `SchemaName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do esquema.

- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro que contém o esquema.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `ConcurrentModificationException`
- `InternalServiceException`

CheckSchemaVersionValidity ação (Python: check_schema_version_validity)

Valida o esquema fornecido. Essa chamada não tem efeitos colaterais, ela simplesmente valida usando o esquema fornecido que usa `DataFormat` como o formato. Como ela não usa um nome de conjunto de esquemas, nenhuma verificação de compatibilidade é realizada.

Solicitação

- `DataFormat` – Obrigatório: string UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  O formato de dados da definição do esquema. No momento, há suporte a `AVRO`, `JSON` e `PROTOBUF`.

- `SchemaDefinition`: obrigatório: string UTF-8, não menos do que 1 ou superior a 170.000 bytes de comprimento, correspondente a Custom string pattern #32.

  A definição do esquema que deve ser validado.

Resposta

- `Valid` – Booleano.

  Retorna true, se o esquema for válido, e false, caso contrário.

- `Error`: string UTF-8, não menos do que 1 ou superior a 5.000 bytes de comprimento.

  Uma mensagem de erro de falha de validação.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

UpdateRegistry ação (Python: update_registry)

Atualiza um registro existente que é usado para armazenar uma coleção de esquemas. As propriedades atualizadas se relacionam com o registro e não modificam nenhum dos esquemas dentro dele.

Solicitação

- `RegistryId` – Obrigatório: um objeto RegistryId.

  Esta é uma estrutura de wrapper que pode conter o nome de registro e o nome do recurso da Amazon (ARN).

- `Description`: obrigatório: string de descrição, inferior a 2.048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do registro. Se a descrição não for fornecida, esse campo não será atualizado.

Resposta

- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro atualizado.

- `RegistryArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome do recurso da Amazon (ARN) do registro atualizado.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `ConcurrentModificationException`
- `InternalServiceException`

GetSchemaByDefinition ação (Python: get_schema_by_definition)

Recupera um esquema pela `SchemaDefinition`. A definição de esquema é enviada para o registro de esquema, canonizada e com hash. Se o hash for correspondido dentro do escopo do `SchemaName` ou ARN (ou o registro padrão, se nenhum for fornecido), os metadados desse esquema serão retornados. Caso contrário, um erro 404 ou NotFound erro será retornado. As versões do esquema no status `Deleted` não serão incluídas nos resultados.

Solicitação

- `SchemaId` – Obrigatório: um objeto SchemaId.

  É uma estrutura de wrapper para conter campos de identidade de esquema. A estrutura contém:

  - SchemaId$SchemaArn: O nome de recurso da Amazon (ARN) do esquema. Um de `SchemaArn` ou `SchemaName` tem de ser fornecido.
  - SchemaId$SchemaName: O nome do esquema. Um de `SchemaArn` ou `SchemaName` tem de ser fornecido.

- `SchemaDefinition`: obrigatório: string UTF-8, não menos do que 1 ou superior a 170.000 bytes de comprimento, correspondente a Custom string pattern #32.

  A definição do esquema para a qual os detalhes do esquema são necessários.

Resposta

- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O ID do esquema da versão do esquema.
- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do esquema.
- `DataFormat` – String UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  O formato de dados da definição do esquema. No momento, há suporte a `AVRO`, `JSON` e `PROTOBUF`.
- `Status` – String UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  O status da versão do esquema.
- `CreatedTime` – String UTF-8.

  A data e a hora em que o esquema foi criado.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetRegistry ação (Python: get_registry)

Descreve o registro especificado em detalhes.

Solicitação

- `RegistryId` – Obrigatório: um objeto RegistryId.

  Esta é uma estrutura de wrapper que pode conter o nome de registro e o nome do recurso da Amazon (ARN).

Resposta

- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro.

- `RegistryArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do registro.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do registro.

- `Status`: string UTF-8 (valores válidos: `AVAILABLE` | `DELETING`).

  O status do registro.

- `CreatedTime` – String UTF-8.

  A data e a hora em que o registro foi criado.

- `UpdatedTime` – String UTF-8.

  A data e a hora em que o registro foi atualizado.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

PutSchemaVersionMetadata ação (Python: put_schema_version_metadata)

Insere o par de chave-valor de metadados para um ID de versão de esquema especificado. Um máximo de dez pares de chave-valor será permitido por versão do esquema. Eles podem ser adicionados em uma ou mais chamadas.

Solicitação

- `SchemaId` – Um objeto SchemaId.

  O ID exclusivo do esquema.
- `SchemaVersionNumber` – Um objeto SchemaVersionNumber.

  O número da versão do esquema.
- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O ID de versão exclusivo da versão do esquema.
- `MetadataKeyValue` – Obrigatório: um objeto MetadataKeyValuePair.

  O valor correspondente da chave de metadados.

Resposta

- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome do recurso da Amazon (ARN) do esquema.
- `SchemaName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do esquema.
- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro.
- `LatestVersion` – Booleano.

  A versão mais recente do esquema.

- `VersionNumber`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  O número da versão do esquema.

- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O ID de versão exclusivo da versão do esquema.

- `MetadataKey`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #33.

  A chave de metadados.

- `MetadataValue`: string UTF-8, não menos do que 1 ou superior a 256 bytes de comprimento, correspondente a Custom string pattern #33.

  O valor da chave de metadados.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`

QuerySchemaVersionMetadata ação (Python: query_schema_version_metadata)

Consultas para obter as informações de metadados da versão do esquema.

Solicitação

- `SchemaId` – Um objeto SchemaId.

  Uma estrutura de wrapper que pode conter o nome do esquema e o nome do recurso da Amazon (ARN).

- `SchemaVersionNumber` – Um objeto SchemaVersionNumber.

O número da versão do esquema.

- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O ID de versão exclusivo da versão do esquema.

- `MetadataList` – Uma matriz de objetos MetadataKeyValuePair.

  Pesquisa os pares de chave-valor para metadados. Se eles não forem fornecidos, todas as informações de metadados serão buscadas.

- `MaxResults`: número (inteiro), não menos do que 1 ou superior a 50.

  O número máximo de resultados exigidos por página. Se o valor não for fornecido, esse valor será definido por padrão como 25 por página.

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma chamada de continuação.

Resposta

- `MetadataInfoMap` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes, correspondente a Custom string pattern #33.

  Cada valor é um objeto MetadataInfo A.

  Um mapa de uma chave de metadados e valores associados.

- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O ID de versão exclusivo da versão do esquema.

- `NextToken` – String UTF-8.

  Um token de continuação para paginação da lista de tokens retornada, retornado se o segmento atual da lista não for o último.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`

RemoveSchemaVersionMetadata ação (Python: remove_schema_version_metadata)

Remove um par de chave-valor dos metadados de versão do esquema para o ID de versão do esquema especificado.

Solicitação

- `SchemaId` – Um objeto SchemaId.

  Uma estrutura de wrapper que pode conter o nome do esquema e o nome do recurso da Amazon (ARN).
- `SchemaVersionNumber` – Um objeto SchemaVersionNumber.

  O número da versão do esquema.
- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O ID de versão exclusivo da versão do esquema.
- `MetadataKeyValue` – Obrigatório: um objeto MetadataKeyValuePair.

  O valor da chave de metadados.

Resposta

- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do esquema.
- `SchemaName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

O nome do esquema.

- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro.

- `LatestVersion` – Booleano.

  A versão mais recente do esquema.

- `VersionNumber`: número (inteiro longo), não menos do que 1 ou superior a 100.000.

  O número da versão do esquema.

- `SchemaVersionId`: string UTF-8, não menos do que 36 ou superior a 36 bytes de comprimento, correspondente a Custom string pattern #17.

  O ID da versão para a versão do esquema.

- `MetadataKey`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #33.

  A chave de metadados.

- `MetadataValue`: string UTF-8, não menos do que 1 ou superior a 256 bytes de comprimento, correspondente a Custom string pattern #33.

  O valor da chave de metadados.

Erros

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`

DeleteRegistry ação (Python: delete_registry)

Exclui todo o registro, incluindo esquema e todas as suas versões. Para obter o status da operação de exclusão, é possível chamar a API `GetRegistry` após a chamada assíncrona. Excluir um registro desativará todas as operações online para ele, como as APIs `UpdateRegistry`, `CreateSchema`, `UpdateSchema` e `RegisterSchemaVersion`.

Solicitação

- `RegistryId` – Obrigatório: um objeto RegistryId.

  Esta é uma estrutura de wrapper que pode conter o nome de registro e o nome do recurso da Amazon (ARN).

Resposta

- `RegistryName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do registro sendo excluído.
- `RegistryArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome do recurso da Amazon (ARN) do registro a ser excluído.
- `Status`: string UTF-8 (valores válidos: `AVAILABLE` | `DELETING`).

  O status do registro. Uma operação bem-sucedida retornará o status `Deleting`.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `ConcurrentModificationException`

DeleteSchema ação (Python: delete_schema)

Exclui todo o conjunto de esquemas, incluindo o conjunto de esquemas e todas as suas versões. Para obter o status da operação de exclusão, é possível chamar a API `GetSchema` após a chamada assíncrona. Excluir um registro desativará todas as operações online para ele, como as APIs `GetSchemaByDefinition` e `RegisterSchemaVersion`.

Solicitação

- `SchemaId` – Obrigatório: um objeto SchemaId.

Esta é uma estrutura de wrapper que pode conter o nome do esquema e o nome do recurso da Amazon (ARN).

Resposta

- `SchemaArn`: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome do recurso da Amazon (ARN) do esquema sendo excluído.
- `SchemaName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #18.

  O nome do esquema sendo excluído.
- `Status` – String UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `DELETING`).

  O status do esquema.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `ConcurrentModificationException`

DeleteSchemaVersions ação (Python: delete_schema_versions)

Remove versões do esquema especificado. Um número de versão ou intervalo pode ser fornecido. Se o modo de compatibilidade proibir a exclusão de uma versão que é necessária, como BACKWARDS_FULL, um erro será retornado. Chamar a API `GetSchemaVersions` após essa chamada listará o status das versões excluídas.

Quando o intervalo de números de versão contiver a versão de ponto de verificação, a API retornará um conflito 409 e não prosseguirá com a exclusão. Primeiro, é preciso remover o ponto de verificação usando a API `DeleteSchemaCheckpoint` antes de usar essa API.

Não é possível utilizar a API `DeleteSchemaVersions` para excluir a primeira versão do esquema no conjunto de esquemas. A primeira versão do esquema só pode ser excluída pela API `DeleteSchema`. Essa operação também excluirá os `SchemaVersionMetadata` anexados nas versões do esquema. Exclusões irreversíveis serão impostas no banco de dados.

Se o modo de compatibilidade proibir a exclusão de uma versão que é necessária, como BACKWARDS_FULL, um erro será retornado.

Solicitação

- `SchemaId` – Obrigatório: um objeto SchemaId.

  Esta é uma estrutura de wrapper que pode conter o nome do esquema e o nome do recurso da Amazon (ARN).

- `Versions`: obrigatório: string UTF-8, não menos do que 1 ou superior a 100.000 bytes de comprimento, correspondente a Custom string pattern #34.

  Pode ser fornecido um intervalo de versões que pode ser do formato:

  - um único número de versão, 5
  - um intervalo, 5-8: exclui as versões 5, 6, 7 e 8

Resposta

- `SchemaVersionErrors` – Uma matriz de objetos SchemaVersionErrorItem.

  Uma lista de objetos `SchemaVersionErrorItem`, cada um contendo uma versão de esquema e erro.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `ConcurrentModificationException`

Fluxos de trabalho

A API de fluxos de trabalho descreve os tipos de dados e a API relacionados à criação, atualização ou exibição de fluxos de trabalho no AWS Glue. O histórico de execução de trabalhos pode ser acessado por 90 dias para seu fluxo de trabalho e execução de trabalhos.

Tipos de dados

- Estrutura JobNodeDetails
- Estrutura CrawlerNodeDetails
- Estrutura TriggerNodeDetails
- Estrutura Crawl
- Estrutura Node
- Estrutura Edge
- Estrutura Workflow
- Estrutura WorkflowGraph
- Estrutura WorkflowRun
- Estrutura WorkflowRunStatistics
- Estrutura StartingEventBatchCondition
- Estrutura Blueprint
- Estrutura BlueprintDetails
- Estrutura LastActiveDefinition
- Estrutura BlueprintRun

Estrutura JobNodeDetails

Os detalhes de um nó de tarefa presente no fluxo de trabalho.

Campos

- `JobRuns` – Uma matriz de objetos JobRun.

  As informações da execução da tarefa representada pelo nó de tarefa.

Estrutura CrawlerNodeDetails

Os detalhes de um nó de crawler presente no fluxo de trabalho.

Campos

- `Crawls` – Uma matriz de objetos Crawl.

  Uma lista de monitoramentos representados pelo nó de monitoramento.

Estrutura TriggerNodeDetails

Os detalhes de um nó de gatilho presente no fluxo de trabalho.

Campos

- `Trigger` – Um objeto Trigger.

  As informações do gatilho representado pelo nó de gatilho.

Estrutura Crawl

Os detalhes de um monitoramento no fluxo de trabalho

Campos

- `State` – String UTF-8 (valores válidos: `RUNNING` | `CANCELLING` | `CANCELLED` | `SUCCEEDED` | `FAILED` | `ERROR`).

  O estado do crawler.
- `StartedOn` – Timestamp.

  A data e a hora em que o monitoramento foi iniciado.
- `CompletedOn` – Timestamp.

  A data e a hora em que o monitoramento foi concluído.
- `ErrorMessage` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A mensagem de erro associada ao crawl.

- `LogGroup` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Log group string pattern.

  O grupo de logs associado ao crawl.

- `LogStream` – String UTF-8, superior a 1 e inferior a 512 bytes de comprimento, correspondente a Log-stream string pattern.

  O fluxo de logs associado ao rastreamento.

Estrutura Node

Um nó representa um componente AWS Glue (acionador, crawler ou trabalho) em um gráfico de fluxo de trabalho.

Campos

- `Type` – String UTF-8 (valores válidos: `CRAWLER` | `JOB` | `TRIGGER`).

  O tipo de componente do AWS Glue representado pelo nó.

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do componente do AWS Glue representado pelo nó.

- `UniqueId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID exclusivo atribuído ao nó no fluxo de trabalho.

- `TriggerDetails` – Um objeto TriggerNodeDetails.

  Detalhes do gatilho quando o nó representa um gatilho.

- `JobDetails` – Um objeto JobNodeDetails.

  Detalhes da tarefa quando o nó representa uma tarefa.

- `CrawlerDetails` – Um objeto CrawlerNodeDetails.

  Detalhes do crawler quando o nó representa um crawler.

Estrutura Edge

Uma borda representa uma conexão direta entre dois componentes do AWS Glue que fazem parte do fluxo de trabalho ao qual a borda pertence.

Campos

- `SourceId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O exclusivo do nó no fluxo de trabalho onde a borda é iniciada.
- `DestinationId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O exclusivo do nó no fluxo de trabalho onde a borda é finalizada.

Estrutura Workflow

Um fluxo de trabalho é uma colação de vários trabalhos e crawlers do AWS Glue dependentes, que são executados para concluir uma tarefa de ETL complexa. Um fluxo de trabalho gerencia a execução e o monitoramento de todos os seus trabalhos e crawlers.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do fluxo de trabalho.
- `Description` – String UTF-8.

  Uma descrição do fluxo de trabalho.
- `DefaultRunProperties` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é uma sequência de caracteres UTF-8.

Um conjunto de propriedades a serem usadas como parte de cada execução do fluxo de trabalho. As propriedades de execução são disponibilizadas para cada trabalho no fluxo de trabalho. Um trabalho pode modificar as propriedades dos próximos trabalhos no fluxo.

- `CreatedOn` – Timestamp.

  A data e a hora em que o fluxo de trabalho foi criado.

- `LastModifiedOn` – Timestamp.

  A data e a hora em que o fluxo de trabalho foi modificado pela última vez.

- `LastRun` – Um objeto WorkflowRun.

  As informações sobre a última execução do fluxo de trabalho.

- `Graph` – Um objeto WorkflowGraph.

  O gráfico que representa todos os componentes do AWS Glue que pertencem ao fluxo de trabalho como nós e conexões direcionadas entre eles como bordas.

- `CreationStatus` – String UTF-8 (valores válidos: `CREATING` | `CREATED` | `CREATION_FAILED`).

  O status de criação do fluxo de trabalho.

- `MaxConcurrentRuns` – Número (íntegro).

  Você pode usar esse parâmetro para evitar várias atualizações indesejadas de dados, para controlar custos ou, em alguns casos, evitar que exceda o número máximo de execuções simultâneas de qualquer um dos trabalhos do componente. Se você deixar esse parâmetro em branco, não haverá limite para o número de execuções de fluxo de trabalho simultâneas.

- `BlueprintDetails` – Um objeto BlueprintDetails.

  Essa estrutura indica os detalhes do blueprint do qual esse fluxo de trabalho específico foi criado.

Estrutura WorkflowGraph

Um gráfico de fluxo de trabalho representa o fluxo de trabalho completo que contém todos os componentes do AWS Glue presentes no fluxo de trabalho e todas as conexões direcionadas entre eles.

Campos

- `Nodes` – Uma matriz de objetos Nó.

  Uma lista dos componentes do AWS Glue que pertencem ao fluxo de trabalho representados como nós.

- `Edges` – Uma matriz de objetos Borda.

  Uma lista de todas as conexões direcionadas entre os nós que pertencem ao fluxo de trabalho.

Estrutura WorkflowRun

Uma execução de fluxo de trabalho é uma execução de um fluxo de trabalho que fornece todas as informações do runtime.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do fluxo de trabalho que foi executado.

- `WorkflowRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID dessa execução de fluxo de trabalho.

- `PreviousRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da execução do fluxo de trabalho anterior.

- `WorkflowRunProperties` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é uma sequência de caracteres UTF-8.

  As propriedades da execução do fluxo de trabalho que foram definidas durante a execução.

- `StartedOn` – Timestamp.

A data e a hora em que a execução do fluxo de trabalho foi iniciada.

- `CompletedOn` – Timestamp.

  A data e a hora em que a execução do fluxo de trabalho foi concluída.

- `Status` – String UTF-8 (valores válidos: `RUNNING` | `COMPLETED` | `STOPPING` | `STOPPED` | `ERROR`).

  O status da execução do fluxo de trabalho.

- `ErrorMessage` – String UTF-8.

  Essa mensagem de erro descreve qualquer erro que possa ter ocorrido ao iniciar a execução do fluxo de trabalho. Atualmente, a única mensagem de erro é “Concurrent runs exceeded for workflow: `foo`” (Execuções simultâneas excedidas para fluxo de trabalho: foo).

- `Statistics` – Um objeto WorkflowRunStatistics.

  As estatísticas da execução.

- `Graph` – Um objeto WorkflowGraph.

  O gráfico que representa todos os componentes do AWS Glue que pertencem ao fluxo de trabalho como nós e conexões direcionadas entre eles como bordas.

- `StartingEventBatchCondition` – Um objeto StartingEventBatchCondition.

  A condição do lote que iniciou a execução do fluxo de trabalho.

Estrutura WorkflowRunStatistics

As estatísticas da execução do fluxo de trabalho fornecem estatísticas sobre a execução do fluxo de trabalho.

Campos

- `TotalActions` – Número (íntegro).

  O número total de ações na execução do fluxo de trabalho.

- `TimeoutActions` – Número (íntegro).

  O número total de ações que atingiram o tempo limite.

- `FailedActions` – Número (íntegro).

O número total de ações que falharam.

- `StoppedActions` – Número (íntegro).

  O número total de ações que foram interrompidas.

- `SucceededActions` – Número (íntegro).

  O número total de ações que foram bem-sucedidas.

- `RunningActions` – Número (íntegro).

  O número total de ações no estado de execução.

- `ErroredActions` – Número (íntegro).

  Indica a contagem de execuções de trabalho no estado ERROR (ERRO) na execução do fluxo de trabalho.

- `WaitingActions` – Número (íntegro).

  Indica a contagem de execuções de trabalho no estado WAITING (AGUARDANDO) na execução do fluxo de trabalho.

Estrutura StartingEventBatchCondition

A condição do lote que iniciou a execução do fluxo de trabalho. Ou o número de eventos no tamanho do lote chegou, caso em que o membro BatchSize é diferente de zero, ou a janela do lote expirou, caso em que o membro BatchWindow é diferente de zero.

Campos

- `BatchSize` – Número (íntegro).

  Número de eventos no lote.

- `BatchWindow` – Número (íntegro).

  Duração da janela do lote em segundos.

Estrutura Blueprint

Os detalhes de um blueprint.

Campos

- `Name`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  O nome do blueprint.

- `Description`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

  A descrição do blueprint.

- `CreatedOn` – Timestamp.

  A data e a hora em que o blueprint foi registrado.

- `LastModifiedOn` – Timestamp.

  A data e a hora em que o blueprint foi modificado pela última vez.

- `ParameterSpec`: string UTF-8, não menos do que 1 ou superior a 131.072 bytes de comprimento.

  Uma string JSON que indica a lista de especificações de parâmetros para o blueprint.

- `BlueprintLocation` – String UTF-8.

  Especifica o caminho no Amazon S3 onde o blueprint é publicado.

- `BlueprintServiceLocation` – String UTF-8.

  Especifica um caminho no Amazon S3 onde o blueprint é copiado quando você chama `CreateBlueprint/UpdateBlueprint` para registrá-lo no AWS Glue.

- `Status` – String UTF-8 (valores válidos: `CREATING` | `ACTIVE` | `UPDATING` | `FAILED`).

  O status do registro do blueprint.

  - Creating (Criando): o registro do blueprint está em andamento.
  - Active (Ativo): o blueprint foi registrado com sucesso.
  - Updating (Atualizando): uma atualização para o registro do blueprint está em andamento.
  - Failed (Falha): falha no registro do blueprint.

- `ErrorMessage` – String UTF-8.

  Uma mensagem de erro.

- `LastActiveDefinition` – Um objeto LastActiveDefinition.

Quando há várias versões de um blueprint e a versão mais recente apresenta alguns erros, esse atributo indica a última definição do blueprint bem-sucedida que está disponível com o serviço.

Estrutura BlueprintDetails

Os detalhes de um blueprint.

Campos

- `BlueprintName`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  O nome do blueprint.
- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de execução desse blueprint.

Estrutura LastActiveDefinition

Quando há várias versões de um blueprint e a versão mais recente apresenta alguns erros, esse atributo indica a última definição do blueprint bem-sucedida que está disponível com o serviço.

Campos

- `Description`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

  A descrição do blueprint.
- `LastModifiedOn` – Timestamp.

  A data e a hora em que o blueprint foi modificado pela última vez.
- `ParameterSpec`: string UTF-8, não menos do que 1 ou superior a 131.072 bytes de comprimento.

  Uma string JSON que especifica os parâmetros para o blueprint.
- `BlueprintLocation` – String UTF-8.

  Especifica um caminho no Amazon S3 em que o blueprint é publicado pelo desenvolvedor do AWS Glue.

- `BlueprintServiceLocation` – String UTF-8.

  Especifica um caminho no Amazon S3 onde o esquema é copiado quando você cria ou atualiza o esquema.

Estrutura BlueprintRun

Os detalhes de uma execução de blueprint.

Campos

- `BlueprintName`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  O nome do blueprint.

- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de execução para essa execução de blueprint.

- `WorkflowName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de um fluxo de trabalho criado como resultado de uma execução bem-sucedida do blueprint. Se uma execução de blueprint tiver um erro, não haverá um fluxo de trabalho criado.

- `State` – String UTF-8 (valores válidos: `RUNNING` | `SUCCEEDED` | `FAILED` | `ROLLING_BACK`).

  O estado da execução do blueprint. Os valores possíveis são:

  - Running (Executando): a execução do blueprint está em andamento.
  - Succeeded (Êxito): a execução do blueprint foi concluída com êxito.
  - Failed (Falha): a execução do blueprint falhou e a reversão está concluída.
  - Rolling Back (Revertendo): a execução do blueprint falhou e a reversão está em andamento.

- `StartedOn` – Timestamp.

  A data e a hora em que a execução do blueprint foi iniciada.

- `CompletedOn` – Timestamp.

  A data e a hora em que a execução do blueprint foi concluída.

- `ErrorMessage` – String UTF-8.

  Indica os erros que são vistos durante a execução do blueprint.

- `RollbackErrorMessage` – String UTF-8.

  Se houver erros ao criar as entidades de um fluxo de trabalho, tentamos reverter as entidades criadas até esse ponto e excluí-las. Esse atributo indica os erros vistos ao tentar excluir as entidades que foram criadas.

- `Parameters`: string UTF-8, não menos do que 1 ou superior a 131.072 bytes de comprimento.

  Os parâmetros do blueprint como uma string. Você terá que fornecer um valor para cada chave exigida pela especificação do parâmetro, que é definida em `Blueprint$ParameterSpec`.

- `RoleArn` – String UTF-8, superior a 1 e inferior a 1024 bytes de comprimento, correspondente a Custom string pattern #26.

  O ARN da função. Essa função será assumida pelo produto AWS Glue e será usada para criar o fluxo de trabalho e outras entidades de um fluxo de trabalho.

Operações

- Ação CreateWorkflow (Python: create_workflow)
- Ação UpdateWorkflow (Python: update_workflow)
- Ação DeleteWorkflow (Python: delete_workflow)
- Ação GetWorkflow (Python: get_workflow)
- Ação ListWorkflows (Python: list_workflows)
- Ação BatchGetWorkflows (Python: batch_get_workflows)
- Ação GetWorkflowRun (Python: get_workflow_run)
- Ação GetWorkflowRuns (Python: get_workflow_runs)
- Ação GetWorkflowRunProperties (Python: get_workflow_run_properties)
- Ação PutWorkflowRunProperties (Python: put_workflow_run_properties)
- Ação CreateBlueprint (Python: create_blueprint)
- Ação UpdateBlueprint (Python: update_blueprint)
- Ação DeleteBlueprint (Python: delete_blueprint)
- Ação ListBlueprints (Python: list_blueprints)

- Ação BatchGetBlueprints (Python: batch_get_blueprints)
- Ação StartBlueprintRun (Python: start_blueprint_run)
- Ação GetBlueprintRun (Python: get_blueprint_run)
- Ação GetBlueprintRuns (Python: get_blueprint_runs)
- Ação StartWorkflowRun (Python: start_workflow_run)
- Ação StopWorkflowRun (Python: stop_workflow_run)
- Ação ResumeWorkflowRun (Python: resume_workflow_run)

Ação CreateWorkflow (Python: create_workflow)

Cria uma nova workload.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome a ser atribuído ao fluxo de trabalho. Ele deve ser exclusivo na conta.
- `Description` – String UTF-8.

  Uma descrição do fluxo de trabalho.
- `DefaultRunProperties` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é uma sequência de caracteres UTF-8.

  Um conjunto de propriedades a serem usadas como parte de cada execução do fluxo de trabalho.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags a serem usadas com esse fluxo de trabalho.
- `MaxConcurrentRuns` – Número (íntegro).

Você pode usar esse parâmetro para evitar várias atualizações indesejadas de dados, para controlar custos ou, em alguns casos, evitar que exceda o número máximo de execuções simultâneas de qualquer um dos trabalhos do componente. Se você deixar esse parâmetro em branco, não haverá limite para o número de execuções de fluxo de trabalho simultâneas.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do fluxo de trabalho que foi fornecido como parte da solicitação.

Erros

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

Ação UpdateWorkflow (Python: update_workflow)

Atualiza um fluxo de trabalho existente.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do fluxo de trabalho a ser atualizado.
- `Description` – String UTF-8.

  A descrição do fluxo de trabalho.
- `DefaultRunProperties` – Um array de mapa dos pares de valor-chave.

Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

Cada valor é uma sequência de caracteres UTF-8.

Um conjunto de propriedades a serem usadas como parte de cada execução do fluxo de trabalho.

- `MaxConcurrentRuns` – Número (íntegro).

  Você pode usar esse parâmetro para evitar várias atualizações indesejadas de dados, para controlar custos ou, em alguns casos, evitar que exceda o número máximo de execuções simultâneas de qualquer um dos trabalhos do componente. Se você deixar esse parâmetro em branco, não haverá limite para o número de execuções de fluxo de trabalho simultâneas.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do fluxo de trabalho que foi especificado na entrada.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Ação DeleteWorkflow (Python: delete_workflow)

Exclui um fluxo de trabalho.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

Nome do fluxo de trabalho a ser excluído.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do fluxo de trabalho especificado na entrada.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Ação GetWorkflow (Python: get_workflow)

Recupera metadados de recurso para um fluxo de trabalho.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do fluxo de trabalho a ser recuperado.
- `IncludeGraph` – Booleano.

  Especifica se um gráfico será incluído ao retornar os metadados do recurso do fluxo de trabalho.

Resposta

- `Workflow` – Um objeto Fluxo de trabalho.

  Os metadados de recurso para o fluxo de trabalho.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação ListWorkflows (Python: list_workflows)

Lista nomes de fluxos de trabalho criados na conta.

Solicitação

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma solicitação de continuação.
- `MaxResults`: número (inteiro), não inferior a 1 nem superior a 25.

  O tamanho máximo de uma lista a ser retornada.

Resposta

- `Workflows`: uma matriz de strings UTF-8, não menos do que 1 ou superior a 25 strings.

  Lista de nomes de fluxos de trabalho na conta.
- `NextToken` – String UTF-8.

  Um token de continuação, caso nem todos os nomes de fluxos de trabalho tenham sido retornados.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação BatchGetWorkflows (Python: batch_get_workflows)

Retorna uma lista de metadados de recurso para uma lista de nomes de fluxos de trabalho. Depois de chamar a operação `ListWorkflows`, você pode chamar essa operação para acessar os dados aos quais você recebeu permissões. Essa operação oferece suporte a todas as permissões do IAM, incluindo condições de permissão que usam tags.

Solicitação

- `Names` – Obrigatório: uma matriz de strings UTF-8, no mínimo 1 ou mais de 25 strings.

  Uma lista de nomes de fluxos de trabalho, que podem ser os nomes retornados pela operação `ListWorkflows`.
- `IncludeGraph` – Booleano.

  Especifica se um gráfico será incluído ao retornar os metadados do recurso do fluxo de trabalho.

Resposta

- `Workflows`: uma matriz de objetos Fluxo de trabalho, não menos do que 1 ou superior a 25 estruturas.

  Uma lista de metadados do recurso do fluxo de trabalho.
- `MissingWorkflows`: uma matriz de strings UTF-8, não menos do que 1 ou superior a 25 strings.

  Uma lista de nomes de fluxos de trabalho não encontrados.

Erros

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Ação GetWorkflowRun (Python: get_workflow_run)

Recupera os metadados de determinada execução do fluxo de trabalho. O histórico de execução de trabalhos pode ser acessado por 90 dias para seu fluxo de trabalho e execução de trabalhos.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do fluxo de trabalho em execução.
- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da execução do fluxo de trabalho.
- `IncludeGraph` – Booleano.

  Especifica se deve incluir o gráfico do fluxo de trabalho como resposta ou não.

Resposta

- `Run` – Um objeto WorkflowRun.

  Os metadados da execução do fluxo de trabalho solicitados.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetWorkflowRuns (Python: get_workflow_runs)

Recupera os metadados de todas as execuções de determinado fluxo de trabalho.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do fluxo de trabalho cujos metadados de execuções devem ser retornados.
- `IncludeGraph` – Booleano.

Especifica se deve incluir o gráfico do fluxo de trabalho como resposta ou não.

- `NextToken` – String UTF-8.

  O tamanho máximo da resposta.

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de execuções do fluxo de trabalho a serem incluídas na resposta.

Resposta

- `Runs`: uma matriz de objetos WorkflowRun, não menos do que 1 ou superior a 1.000 estruturas.

  Uma lista de objetos de metadados da execução do fluxo de trabalho.

- `NextToken` – String UTF-8.

  Um token de continuação, se todas as execuções do fluxo de trabalho solicitadas não tiverem sido retornadas.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetWorkflowRunProperties (Python: get_workflow_run_properties)

Recupera as propriedades de execução do fluxo de trabalho que foram definidas durante a execução.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do fluxo de trabalho que foi executado.

- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da execução do fluxo de trabalho cujas propriedades deveriam ser retornadas.

Resposta

- `RunProperties` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é uma sequência de caracteres UTF-8.

  As propriedades da execução do fluxo de trabalho que foram definidas durante a execução especificada.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação PutWorkflowRunProperties (Python: put_workflow_run_properties)

Coloca as propriedades da execução do fluxo de trabalho especificadas para determinada execução do fluxo de trabalho. Se uma propriedade já existir para a execução especificada, ela substituirá o valor. Caso contrário, a propriedade será adicionada às propriedades existentes.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome do fluxo de trabalho que foi executado.
- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O ID da execução do fluxo de trabalho para o qual as propriedades de execução deveriam ser atualizadas.

- `RunProperties`: obrigatório: uma matriz de mapa dos pares de chave-valor.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é uma sequência de caracteres UTF-8.

  As propriedades para colocar a execução especificada.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `AlreadyExistsException`
- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

Ação CreateBlueprint (Python: create_blueprint)

Regista um blueprint com o AWS Glue.

Solicitação

- `Name`: obrigatório: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  O nome do blueprint.

- `Description`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

Uma descrição do blueprint.

- `BlueprintLocation`: obrigatório: string UTF-8, não menos do que 1 ou superior a 8.192 bytes de comprimento, correspondente a Custom string pattern #28.

  Especifica um caminho no Amazon S3 em que o blueprint é publicado.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags a serem aplicadas a esse blueprint.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Retorna o nome do blueprint que foi registrado.

Erros

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`

Ação UpdateBlueprint (Python: update_blueprint)

Atualiza um blueprint registrado.

Solicitação

- `Name`: obrigatório: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

O nome do blueprint.

- `Description`: string UTF-8, não menos do que 1 ou superior a 512 bytes de comprimento.

  Uma descrição do blueprint.

- `BlueprintLocation`: obrigatório: string UTF-8, não menos do que 1 ou superior a 8.192 bytes de comprimento, correspondente a Custom string pattern #28.

  Especifica um caminho no Amazon S3 em que o blueprint é publicado.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Retorna o nome do blueprint que foi atualizado.

Erros

- `EntityNotFoundException`
- `ConcurrentModificationException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `IllegalBlueprintStateException`

Ação DeleteBlueprint (Python: delete_blueprint)

Exclui um blueprint existente.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do blueprint a ser excluído.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Retorna o nome do blueprint que foi excluído.

Erros

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Ação ListBlueprints (Python: list_blueprints)

Lista todos os nomes de blueprints em uma conta.

Solicitação

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma solicitação de continuação.
- `MaxResults`: número (inteiro), não inferior a 1 nem superior a 25.

  O tamanho máximo de uma lista a ser retornada.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  Filtra a lista por uma tag de recurso da AWS.

Resposta

- `Blueprints` – Uma matriz de strings UTF-8.

  Lista de nomes de blueprints na conta.
- `NextToken` – String UTF-8.

Um token de continuação, se nem todos os nomes de blueprints sejam retornados.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação BatchGetBlueprints (Python: batch_get_blueprints)

Recupera informações sobre uma lista de blueprints.

Solicitação

- `Names` – Obrigatório: uma matriz de strings UTF-8, no mínimo 1 ou mais de 25 strings.

  Uma lista de nomes de blueprints.
- `IncludeBlueprint` – Booleano.

  Especifica se deve incluir o blueprint na resposta ou não.
- `IncludeParameterSpec` – Booleano.

  Especifica se deve incluir os parâmetros para o blueprint, como uma string JSON, na resposta ou não.

Resposta

- `Blueprints` – Uma matriz de objetos Blueprint.

  Retorna uma lista de blueprints como um objeto `Blueprints`.
- `MissingBlueprints` – Uma matriz de strings UTF-8.

  Retorna uma lista de `BlueprintNames` que não foram encontrados.

Erros

- `InternalServiceException`

- `OperationTimeoutException`
- `InvalidInputException`

Ação StartBlueprintRun (Python: start_blueprint_run)

Inicia uma nova execução do blueprint especificado.

Solicitação

- `BlueprintName`: obrigatório: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  O nome do blueprint.
- `Parameters`: string UTF-8, não menos do que 1 ou superior a 131.072 bytes de comprimento.

  Especifica os parâmetros como um objeto `BlueprintParameters`.
- `RoleArn` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 1024 bytes de comprimento, correspondente a Custom string pattern #26.

  Especifica a função do IAM usada para criar o fluxo de trabalho.

Resposta

- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de execução para essa execução de blueprint.

Erros

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`
- `EntityNotFoundException`
- `IllegalBlueprintStateException`

Ação GetBlueprintRun (Python: get_blueprint_run)

Recupera os detalhes de uma execução de bluetprint.

Solicitação

- `BlueprintName`: obrigatório: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #27.

  O nome do blueprint.
- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de execução do blueprint que você deseja recuperar.

Resposta

- `BlueprintRun` – Um objeto BlueprintRun.

  Informa um objeto `BlueprintRun`.

Erros

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação GetBlueprintRuns (Python: get_blueprint_runs)

Recupera os detalhes das execuções de blueprint para um blueprint especificado.

Solicitação

- `BlueprintName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do blueprint.
- `NextToken` – String UTF-8.

Um token de continuação, se esta for uma solicitação de continuação.

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo de uma lista a ser retornada.

Resposta

- `BlueprintRuns` – Uma matriz de objetos BlueprintRun.

  Retorna uma lista de objetos `BlueprintRun`.
- `NextToken` – String UTF-8.

  Um token de continuação, se nem todas as execuções de blueprint solicitadas forem retornadas.

Erros

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Ação StartWorkflowRun (Python: start_workflow_run)

Inicia uma nova execução do fluxo de trabalho especificado.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do fluxo de trabalho a ser iniciado.
- `RunProperties` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é uma sequência de caracteres UTF-8.

As propriedades de execução de fluxo de trabalho para a nova execução de fluxo de trabalho.

Resposta

- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um ID para a nova execução.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

Ação StopWorkflowRun (Python: stop_workflow_run)

Interrompe a execução do fluxo de trabalho especificada.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do fluxo de trabalho a ser interrompido.
- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da execução do fluxo de trabalho a ser interrompida.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `IllegalWorkflowStateException`

Ação ResumeWorkflowRun (Python: resume_workflow_run)

Reinicia os nós selecionados de uma execução de fluxo de trabalho anterior parcialmente concluída e retoma a execução do fluxo de trabalho. Os nós selecionados e todos os nós downstream a eles são executados.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do fluxo de trabalho a ser retomado.
- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da execução do fluxo de trabalho a ser retomada.
- `NodeIds`: obrigatório: uma matriz de strings UTF-8.

  Uma lista de IDs de nó daqueles que você deseja reiniciar. Os nós que devem ser reiniciados devem ter uma tentativa de execução na execução original.

Resposta

- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O novo ID atribuído à execução do fluxo de trabalho retomada. Cada retomada de uma execução de fluxo de trabalho terá um novo ID de execução.
- `NodeIds` – Uma matriz de strings UTF-8.

Uma lista de IDs dos nós que foram realmente reiniciados.

Erros

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentRunsExceededException`
- `IllegalWorkflowStateException`

Perfis de uso

A API de perfis de uso descreve os tipos de dados e a API relacionados à criação, atualização ou visualização de perfis de uso em AWS Glue.

Tipos de dados

- ProfileConfiguration estrutura
- ConfigurationObject estrutura
- UsageProfileDefinition estrutura

ProfileConfiguration estrutura

Especifica os valores de trabalho e sessão que um administrador configura em um perfil de AWS Glue uso.

Campos

- `SessionConfiguration` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é um objeto ConfigurationObject A.

Um mapa de valores-chave dos parâmetros de configuração para AWS Glue sessões.

- `JobConfiguration` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é um objeto ConfigurationObject A.

  Um mapa de valores-chave dos parâmetros de configuração para AWS Glue trabalhos.

ConfigurationObject estrutura

Especifica os valores que um administrador define para cada parâmetro de trabalho ou sessão configurado em um perfil de AWS Glue uso.

Campos

- `DefaultValue`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #31.

  Um valor padrão para o parâmetro.
- `AllowedValues` – Uma matriz de strings UTF-8.

  Uma lista de valores permitidos para o parâmetro.
- `MinValue`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #31.

  Um valor mínimo permitido para o parâmetro.
- `MaxValue`: string UTF-8, não menos do que 1 ou superior a 128 bytes de comprimento, correspondente a Custom string pattern #31.

  Um valor máximo permitido para o parâmetro.

UsageProfileDefinition estrutura

Descreve um perfil de AWS Glue uso.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do perfil de uso.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do perfil de uso.
- `CreatedOn` – Timestamp.

  A data e a hora em que o perfil de uso foi criado.
- `LastModifiedOn` – Timestamp.

  A data e a hora em que o perfil de uso foi modificado pela última vez.

Operações

- CreateUsageProfile ação (Python: create_usage_profile)
- GetUsageProfile ação (Python: get_usage_profile)
- UpdateUsageProfile ação (Python: update_usage_profile)
- DeleteUsageProfile ação (Python: delete_usage_profile)
- ListUsageProfiles ação (Python: list_usage_profiles)

CreateUsageProfile ação (Python: create_usage_profile)

Cria um perfil de AWS Glue uso.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do perfil de uso.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

Uma descrição do perfil de uso.

- `Configuration` – Obrigatório: um objeto ProfileConfiguration.

  Um `ProfileConfiguration` objeto que especifica os valores do trabalho e da sessão para o perfil.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  Uma lista de tags aplicadas ao perfil de uso.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do perfil de uso que foi criado.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `AlreadyExistsException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `OperationNotSupportedException`

GetUsageProfile ação (Python: get_usage_profile)

Recupera informações sobre o perfil de AWS Glue uso especificado.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome do perfil de uso a ser recuperado.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do perfil de uso.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do perfil de uso.

- `Configuration` – Um objeto ProfileConfiguration.

  Um `ProfileConfiguration` objeto que especifica os valores do trabalho e da sessão para o perfil.

- `CreatedOn` – Timestamp.

  A data e a hora em que o perfil de uso foi criado.

- `LastModifiedOn` – Timestamp.

  A data e a hora em que o perfil de uso foi modificado pela última vez.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `OperationNotSupportedException`

UpdateUsageProfile ação (Python: update_usage_profile)

Atualize um perfil de AWS Glue uso.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do perfil de uso.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do perfil de uso.

- `Configuration` – Obrigatório: um objeto ProfileConfiguration.

  Um `ProfileConfiguration` objeto que especifica os valores do trabalho e da sessão para o perfil.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do perfil de uso que foi atualizado.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `OperationNotSupportedException`
- `ConcurrentModificationException`

DeleteUsageProfile ação (Python: delete_usage_profile)

Exclui o perfil de uso AWS Glue especificado.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do perfil de uso a ser excluído.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `OperationNotSupportedException`

ListUsageProfiles ação (Python: list_usage_profiles)

Liste todos os perfis AWS Glue de uso.

Solicitação

- `NextToken`: string UTF-8, não superior a 400.000 bytes de comprimento.

  Um token de continuação, incluído se esta for uma chamada de continuação.
- `MaxResults`: número (inteiro) não inferior a 1 nem superior a 200.

  O número máximo de perfis de uso a serem retornados em uma única resposta.

Resposta

- `Profiles` – Uma matriz de objetos UsageProfileDefinition.

  Uma lista de objetos de perfil de uso (`UsageProfileDefinition`).
- `NextToken`: string UTF-8, não superior a 400.000 bytes de comprimento.

Um token de continuação, presente se o segmento de lista atual não for o último.

Erros

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `OperationNotSupportedException`

API de machine learning

A API Machine learning descreve os tipos de dados de machine learning e inclui a API para criar, excluir ou atualizar uma transformação, ou iniciar a execução de uma tarefa de machine learning.

Tipos de dados

- Estrutura TransformParameters
- Estrutura EvaluationMetrics
- Estrutura MLTransform
- Estrutura FindMatchesParameters
- Estrutura FindMatchesMetrics
- Estrutura ConfusionMatrix
- Estrutura GlueTable
- Estrutura TaskRun
- Estrutura TransformFilterCriteria
- Estrutura TransformSortCriteria
- Estrutura TaskRunFilterCriteria
- Estrutura TaskRunSortCriteria
- Estrutura TaskRunProperties
- Estrutura FindMatchesTaskRunProperties
- Estrutura ImportLabelsTaskRunProperties
- Estrutura ExportLabelsTaskRunProperties

- Estrutura LabelingSetGenerationTaskRunProperties
- Estrutura SchemaColumn
- Estrutura TransformEncryption
- Estrutura MLUserDataEncryption
- Estrutura ColumnImportance

Estrutura TransformParameters

Os parâmetros de algoritmo específicos associados à transformação de machine learning.

Campos

- `TransformType`: obrigatório: string UTF-8 (valores válidos: `FIND_MATCHES`).

  O tipo de transformação de machine learning.

  Para obter informações sobre os tipos de transformações de machine learning, consulte Criação de transformações para machine learning.
- `FindMatchesParameters` – Um objeto FindMatchesParameters.

  Os parâmetros do algoritmo de busca de correspondências.

Estrutura EvaluationMetrics

As métricas de avaliação fornecem uma estimativa da qualidade da transformação de machine learning.

Campos

- `TransformType`: obrigatório: string UTF-8 (valores válidos: `FIND_MATCHES`).

  O tipo de transformação de machine learning.
- `FindMatchesMetrics` – Um objeto FindMatchesMetrics.

  As métricas de avaliação do algoritmo de busca de correspondências.

Estrutura MLTransform

Uma estrutura para a transformação de machine learning.

Campos

- `TransformId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de transformação exclusivo gerado para a transformação de machine learning. É garantido que o ID será único e não mudará.

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um nome definido pelo usuário para a transformação de machine learning. Não é garantido que os nomes serão únicos, e poderão ser trocados a qualquer momento.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Um texto de descrição de formato longo e definido pelo usuário para a transformação de machine learning. Não é garantido que as descrições serão únicas, e elas poderão mudar a qualquer momento.

- `Status` – String UTF-8 (valores válidos: `NOT_READY` | `READY` | `DELETING`).

  O status atual da transformação de machine learning.

- `CreatedOn` – Timestamp.

  Um carimbo de data/hora. A data e hora em que essa transformação de machine learning foi criada.

- `LastModifiedOn` – Timestamp.

  Um carimbo de data/hora. O último momento em que essa transformação de machine learning foi modificada.

- `InputRecordTables`: uma matriz de objetos GlueTable, não mais do que dez estruturas.

  Uma lista de definições de tabela do AWS Glue usadas pela transformação.

- `Parameters` – Um objeto TransformParameters.

Um objeto `TransformParameters`. Você pode usar parâmetros para ajustar (personalizar) o comportamento da transformação de machine learning, especificando quais dados ela aprenderá e sua preferência sobre vários tradeoffs (como precisão vs. revocação, ou acurácia vs. custo).

- `EvaluationMetrics` – Um objeto EvaluationMetrics.

  Um objeto `EvaluationMetrics`. As métricas de avaliação fornecem uma estimativa da qualidade da transformação de machine learning.

- `LabelCount` – Número (íntegro).

  Um identificador de contagem para rotular arquivos gerados pelo AWS Glue para essa transformação. À medida que você cria uma transformação melhor, é possível fazer download, rotular e fazer upload do arquivo de rotulamento de maneira interativa.

- `Schema`: uma matriz de objetos SchemaColumn, não mais de 100 estruturas.

  Um mapa de pares de chave-valor representando as colunas e os tipos de dados que podem ser executados em relação a essa transformação. Tem um limite superior de 100 colunas.

- `Role` – String UTF-8.

  O nome ou o nome de recurso da Amazon (ARN) da função do IAM com as permissões necessárias. As permissões necessárias incluem as permissões de função de serviço do AWS Glue para recursos do AWS Glue e as permissões do Amazon S3 exigidas pela transformação.

  - Essa função precisa de permissões de função de serviço do AWS Glue para conceder o acesso aos recursos no AWS Glue. Consulte Anexar uma política aos usuários do IAM que acessam o AWS Glue.
  - Essa função precisa de permissão para suas origens, seus destinos, seu diretório temporário, seus scripts e quaisquer bibliotecas do Amazon Simple Storage Service (Amazon S3) usadas pela execução de tarefa nessa transformação.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  Este valor determina com qual versão de AWS Glue essa transformação de machine learning é compatível. O Glue 1.0 é recomendado para a maioria dos clientes. Se o valor não for definido, a compatibilidade com o Glue 0.9 será definida como padrão. Para obter mais informações, consulte Versões do AWS Glue no guia do desenvolvedor.

- `MaxCapacity` – Número (duplo).

O número de unidades de processamento de dados (DPUs) do AWS Glue alocadas para execuções de tarefas para essa transformação. Você pode alocar de 2 a 100 DPUs. O padrão é 10. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

`MaxCapacity` é uma opção mutuamente exclusiva com `NumberOfWorkers` e `WorkerType`.

- Se `NumberOfWorkers` ou `WorkerType` estiver definido, `MaxCapacity` não poderá ser definido.
- Se `MaxCapacity` estiver definido, nem `NumberOfWorkers` e nem `WorkerType` poderá ser definido.
- Se `WorkerType` estiver definido, `NumberOfWorkers` será necessário (e vice-versa).
- `MaxCapacity` e `NumberOfWorkers` devem ser pelo menos 1.

Quando o campo `WorkerType` é definido como um valor diferente de `Standard`, o campo `MaxCapacity` é definido automaticamente e se torna somente leitura.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido alocado quando uma tarefa dessa transformação é executada. Aceita um valor de Standard (Padrão), G.1X ou G.2X.

  - Para o tipo de operador `Standard`, cada operador fornece 4 vCPU, 16 GB de memória e um disco de 50 GB e 2 executores por operador.
  - Para o tipo de operador `G.1X`, cada operador fornece 4 vCPU, 16 GB de memória e um disco e 64 GB, 1 executor por operador.
  - Para o tipo de operador `G.2X`, cada operador fornece 8 vCPU, 32 GB de memória e um disco de 128 GB e 1 executor por operador.

  `MaxCapacity` é uma opção mutuamente exclusiva com `NumberOfWorkers` e `WorkerType`.

  - Se `NumberOfWorkers` ou `WorkerType` estiver definido, `MaxCapacity` não poderá ser definido.
  - Se `MaxCapacity` estiver definido, nem `NumberOfWorkers` e nem `WorkerType` poderá ser definido.
  - Se `WorkerType` estiver definido, `NumberOfWorkers` será necessário (e vice-versa).
  - `MaxCapacity` e `NumberOfWorkers` devem ser pelo menos 1.

- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de um `workerType` definido alocados quando uma tarefa da transformação é executada.

  Se `WorkerType` estiver definido, `NumberOfWorkers` será necessário (e vice-versa).

- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite em minutos da transformação de machine learning.

- `MaxRetries` – Número (íntegro).

  O número máximo de novas tentativas após uma falha de transformação `MLTaskRun` de machine learning.

- `TransformEncryption` – Um objeto TransformEncryption.

  As configurações de criptografia em repouso da transformação que se aplicam ao acesso aos dados do usuário. As transformações de machine learning podem acessar dados do usuário criptografados no Amazon S3 usando o KMS.

Estrutura FindMatchesParameters

Os parâmetros para configurar a transformação de localização de correspondências.

Campos

- `PrimaryKeyColumnName` – String UTF-8, superior a 1 e inferior a 1024 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de uma coluna que identifica exclusivamente as linhas na tabela de origem. Usado para ajudar a identificar os registros correspondentes.

- `PrecisionRecallTradeoff`: número (double), no máximo 1,0.

  O valor selecionado ao ajustar a transformação para um equilíbrio entre precisão e revocação. Um valor de 0.5 significa que não há preferência; um valor de 1.0 significa um desvio puramente para precisão, e um valor de 0.0 significa um desvio para revocação. Como este é um tradeoff, escolher valores próximos a 1.0 significa uma revocação muito baixa, e escolher valores próximos a 0.0 resulta em uma precisão muito baixa.

A métrica de precisão indica a frequência com que seu modelo está correto ao prever uma correspondência.

A métrica de revocação indica, para uma correspondência real, com que frequência seu modelo prevê a correspondência.

- `AccuracyCostTradeoff`: número (double), no máximo 1,0.

  O valor selecionado ao ajustar sua transformação para um equilíbrio entre acurácia e custo. Um valor de 0.5 significa que o sistema equilibra as preocupações de acurácia e o custo. Um valor de 1.0 significa um desvio puramente para a acurácia, o que normalmente resulta em um custo mais alto, que é, às vezes, substancialmente superior. Um valor de 0.0 significa um desvio puramente para o custo, o que resulta em uma transformação `FindMatches` menos precisa e que é, às vezes, uma acurácia inaceitável.

  A acurácia mede quão bem a transformação encontra verdadeiros positivos e verdadeiros negativos. Aumentar a acurácia exige mais recursos de máquina e eleva os custos. Mas isso também resulta em um aumento da revocação.

  O custo avalia o número de recursos de computação, ou seja, o dinheiro, consumidos para executar a transformação.

- `EnforceProvidedLabels` – Booleano.

  O valor para ativar/desativar a forçar a saída para corresponder aos rótulos fornecidos por usuários. Se o valor for `True`, a transformação `find matches` forçará a saída para corresponder aos rótulos fornecidos. Os resultados substituem os resultados de confluência normal. Se o valor for `False`, a transformação `find matches` não garantirá que todos os rótulos fornecidos serão respeitados, e os resultados contarão com o modelo treinado.

  Observe que definir esse valor como true pode aumentar o tempo de execução do trabalho.

Estrutura FindMatchesMetrics

As métricas de avaliação do algoritmo de busca de correspondências. A qualidade da transformação de machine learning é avaliada fazendo com que sua transformação preveja algumas correspondências e comparando os resultados com correspondências conhecidas do mesmo conjunto de dados. As métricas de qualidade são baseadas em um subconjunto dos seus dados, para que eles não sejam precisos.

Campos

- `AreaUnderPRCurve`: número (double), no máximo 1,0.

  A área sob a curva de precisão/revocação (AUPRC) é um único número único que mede a qualidade geral da transformação, que é independente da escolha feita em precisão vs. revocação. Valores mais altos indicam que você tem um tradeoff de precisão vs. revocação mais atrativo.

  Para obter mais informações, consulte Precisão e revocação na Wikipédia.

- `Precision`: número (double), no máximo 1,0.

  A métrica de precisão indica a frequência com que sua transformação está correta ao prever uma correspondência. Especificamente, avalia quão bem a transformação encontra verdadeiros positivos do total possível de verdadeiros positivos.

  Para obter mais informações, consulte Precisão e revocação na Wikipédia.

- `Recall`: número (double), no máximo 1,0.

  A métrica de revocação indica, para uma correspondência real, com que frequência sua transformação prevê a correspondência. Especificamente, avalia quão bem a transformação encontra verdadeiros positivos do total de registros nos dados de origem.

  Para obter mais informações, consulte Precisão e revocação na Wikipédia.

- `F1`: número (double), no máximo 1,0.

  O F1 máximo indica a acurácia da transformação entre 0 e 1, onde 1 é a melhor acurácia.

  Para obter mais informações, consulte F1 score na Wikipédia.

- `ConfusionMatrix` – Um objeto ConfusionMatrix.

  A matriz de confusão mostra o que sua transformação está prevendo com precisão, e quais tipos de erros está cometendo.

  Para obter mais informações, consulte Confusion Matrix na Wikipédia.

- `ColumnImportances`: uma matriz de objetos ColumnImportance, não mais de 100 estruturas.

  Uma lista de estruturas `ColumnImportance` contendo métricas de importância de coluna, classificadas em ordem de importância decrescente.

Estrutura ConfusionMatrix

A matriz de confusão mostra o que sua transformação está prevendo com precisão, e quais tipos de erros está cometendo.

Para obter mais informações, consulte Confusion Matrix na Wikipédia.

Campos

- `NumTruePositives` – Número (extenso).

  O número de correspondências nos dados que a transformação encontrou corretamente, na matriz de confusão da sua transformação.
- `NumFalsePositives` – Número (extenso).

  O número de não correspondências nos dados que a transformação incorretamente classificou como correspondências, na matriz de confusão da sua transformação.
- `NumTrueNegatives` – Número (extenso).

  O número de não correspondências nos dados que a transformação rejeitou corretamente, na matriz de confusão da sua transformação.
- `NumFalseNegatives` – Número (extenso).

  O número de correspondências nos dados que a transformação não encontrou, na matriz de confusão da sua transformação.

Estrutura GlueTable

O banco de dados e a tabela no AWS Glue Data Catalog usados para os dados de entrada ou saída.

Campos

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados no AWS Glue Data Catalog.
- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome da tabela no AWS Glue Data Catalog.

- `CatalogId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um identificador exclusivo para o AWS Glue Data Catalog.

- `ConnectionName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da conexão com o AWS Glue Data Catalog.

- `AdditionalOptions`: uma matriz de mapas de pares chave-valor, não menos que 1 o não mais que 10 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é uma string de descrição, com não mais de 2048 de comprimento, correspondo a URI address multi-line string pattern.

  Opções adicionais para a tabela. Atualmente, duas chaves são compatíveis:

  - `pushDownPredicate`: para filtrar por partições sem a necessidade de listar e ler todos os arquivos do conjunto de dados.
  - `catalogPartitionPredicate`: para usar a remoção de partições do lado do servidor usando índices de partição no AWS Glue Data Catalog.

Estrutura TaskRun

Os parâmetros de amostragem associados à transformação de machine learning.

Campos

- `TransformId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação

- `TaskRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O identificador exclusivo para a execução dessa tarefa.

- `Status` – String UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  O status atual da execução de tarefa solicitada.

- `LogGroupName` – String UTF-8.

  Os nomes do grupo de log para proteger os registros, associados a essa execução de tarefa.

- `Properties` – Um objeto TaskRunProperties.

  Especifica as propriedades de configuração associadas à execução de tarefa.

- `ErrorString` – String UTF-8.

  A lista de strings de erros associadas a essa execução de tarefa.

- `StartedOn` – Timestamp.

  A data e a hora em que essa execução de tarefa iniciou.

- `LastModifiedOn` – Timestamp.

  O último momento em que a execução de tarefa foi atualizada.

- `CompletedOn` – Timestamp.

  O último momento em que a execução de tarefa foi finalizada.

- `ExecutionTime` – Número (íntegro).

  A quantidade de tempo (em segundos) em que a tarefa executada consumiu recursos.

Estrutura TransformFilterCriteria

Os critérios usado para filtrar as transformações de machine learning.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um nome de transformação exclusivo usado para filtrar as transformações de machine learning.

- `TransformType` – String UTF-8 (valores válidos: `FIND_MATCHES`).

O tipo de transformação de machine learning usado para filtrar as transformações de machine learning.

- `Status` – String UTF-8 (valores válidos: `NOT_READY` | `READY` | `DELETING`).

  Filtra a lista de transformações de machine learning pelo último status conhecido das transformações (para indicar se uma transformação pode ser usada ou não). Um dos "NOT_READY", "READY" ou "DELETING".

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  Este valor determina com qual versão de AWS Glue essa transformação de machine learning é compatível. O Glue 1.0 é recomendado para a maioria dos clientes. Se o valor não for definido, a compatibilidade com o Glue 0.9 será definida como padrão. Para obter mais informações, consulte Versões do AWS Glue no guia do desenvolvedor.

- `CreatedBefore` – Timestamp.

  A data e hora antes de as transformações serem criadas.

- `CreatedAfter` – Timestamp.

  A data e hora depois de as transformações serem criadas.

- `LastModifiedBefore` – Timestamp.

  Filtro da última modificação das transformações antes dessa data.

- `LastModifiedAfter` – Timestamp.

  Filtro da última modificação das transformações depois dessa data.

- `Schema`: uma matriz de objetos SchemaColumn, não mais de 100 estruturas.

  Filtros dos conjuntos de dados com um esquema específico. O objeto `Map<Column, Type>` é uma matriz de pares de chave-valor que representa o esquema que essa transformação aceita, em que `Column` é o nome de uma coluna, e `Type` é o tipo dos dados como um número inteiro ou uma string. Tem um limite superior de 100 colunas.

Estrutura TransformSortCriteria

Os critérios de classificação associados à transformação de machine learning.

Campos

- `Column` – obrigatório: string UTF-8 (valores válidos: `NAME` | `TRANSFORM_TYPE` | `STATUS` | `CREATED` | `LAST_MODIFIED`).

  A coluna a ser usada nos critérios de classificação que estão associados à transformação de machine learning.

- `SortDirection` – Obrigatório: string UTF-8 (valores válidos: `DESCENDING` | `ASCENDING`).

  A direção da classificação a ser usada nos critérios de classificação associados à transformação de machine learning.

Estrutura TaskRunFilterCriteria

Os critérios usados para filtrar a execução de tarefa para a transformação de machine learning.

Campos

- `TaskRunType` – String UTF-8 (valores válidos: `EVALUATION` | `LABELING_SET_GENERATION` | `IMPORT_LABELS` | `EXPORT_LABELS` | `FIND_MATCHES`).

  O tipo da tarefa executada.

- `Status` – String UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  O status atual da tarefa executada.

- `StartedBefore` – Timestamp.

  Filtra as execuções de tarefas iniciadas antes dessa data.

- `StartedAfter` – Timestamp.

  Filtra as execuções de tarefas iniciadas após essa data.

Estrutura TaskRunSortCriteria

Os critérios de classificação usados para classificar a lista de execuções de tarefas para a transformação de machine learning.

Campos

- `Column` – Obrigatório: string UTF-8 (valores válidos: `TASK_RUN_TYPE` | `STATUS` | `STARTED`).

  A coluna que será usada para classificar a lista de execuções de tarefas para a transformação de machine learning.

- `SortDirection` – Obrigatório: string UTF-8 (valores válidos: `DESCENDING` | `ASCENDING`).

  A direção de classificação que será usada para classificar a lista de execuções de tarefas para a transformação de machine learning.

Estrutura TaskRunProperties

As propriedades de configuração para a execução de tarefa.

Campos

- `TaskType` – String UTF-8 (valores válidos: `EVALUATION` | `LABELING_SET_GENERATION` | `IMPORT_LABELS` | `EXPORT_LABELS` | `FIND_MATCHES`).

  O tipo da tarefa executada.

- `ImportLabelsTaskRunProperties` – Um objeto ImportLabelsTaskRunProperties.

  As propriedades de configuração para uma execução de tarefa de rótulos de importação.

- `ExportLabelsTaskRunProperties` – Um objeto ExportLabelsTaskRunProperties.

  As propriedades de configuração para uma execução de tarefa de rótulos de exportação.

- `LabelingSetGenerationTaskRunProperties` – Um objeto LabelingSetGenerationTaskRunProperties.

  As propriedades de configuração para execução de tarefa de geração de conjunto de rótulos.

- `FindMatchesTaskRunProperties` – Um objeto FindMatchesTaskRunProperties.

  As propriedades de configuração de uma execução de tarefa de busca de correspondências.

Estrutura FindMatchesTaskRunProperties

Especifica as propriedades de configuração de uma execução de tarefa de busca de correspondências.

Campos

- `JobId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do trabalho da execução de tarefa de busca de correspondências.
- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome atribuído ao trabalho da execução de tarefa de busca de correspondências.
- `JobRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da execução de trabalho para a execução de tarefa de busca de correspondências

Estrutura ImportLabelsTaskRunProperties

Especifica as propriedades de configuração para uma execução de tarefa de importação de rótulos.

Campos

- `InputS3Path` – String UTF-8.

  O caminho do Amazon Simple Storage Service (Amazon S3) para onde você importará os rótulos.
- `Replace` – Booleano.

  Indica se você deve substituir os rótulos existentes.

Estrutura ExportLabelsTaskRunProperties

Especifica as propriedades de configuração para uma execução de tarefa de exportação de rótulos.

Campos

- `OutputS3Path` – String UTF-8.

O caminho do Amazon Simple Storage Service (Amazon S3) de onde você exportará os rótulos.

Estrutura LabelingSetGenerationTaskRunProperties

Especifica as propriedades de configuração para execução de tarefa de geração de conjunto de rótulos.

Campos

- `OutputS3Path` – String UTF-8.

  O caminho do Amazon Simple Storage Service (Amazon S3) de onde você irá gerar o conjunto de rótulos.

Estrutura SchemaColumn

Um par de chave-valor representando a coluna e o tipo de dado que podem ser executados em relação a essa transformação. O parâmetro `Schema` do `MLTransform` pode conter até 100 dessas estruturas.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 1024 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da coluna.

- `DataType` – String UTF-8 com comprimento não superior a 131.072 bytes, correspondente a Single-line string pattern.

  O tipo de dados na coluna.

Estrutura TransformEncryption

As configurações de criptografia em repouso da transformação que se aplicam ao acesso aos dados do usuário. As transformações de machine learning podem acessar dados do usuário criptografados no Amazon S3 usando o KMS.

Além disso, rótulos importados e transformações treinadas agora podem ser criptografados usando uma chave KMS fornecida pelo cliente.

Campos

- `MlUserDataEncryption` – Um objeto MLUserDataEncryption.

  Um objeto `MLUserDataEncryption` que contém o modo de criptografia e o ID da chave do KMS fornecida pelo cliente.
- `TaskRunSecurityConfigurationName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da configuração de segurança.

Estrutura MLUserDataEncryption

As configurações de criptografia em repouso da transformação que se aplicam ao acesso aos dados do usuário.

Campos

- `MlUserDataEncryptionMode` – Obrigatório: string UTF-8 (valores válidos: `DISABLED` | `SSE-KMS="SSEKMS"`).

  O modo de criptografia aplicado aos dados do usuário. Os valores válidos são:
  - DISABLED: a criptografia está desativada
  - SSEKMS: uso de criptografia no lado do servidor com o AWS Key Management Service (SSE-KMS) para dados do usuário armazenados no Amazon S3.
- `KmsKeyId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da chave do KMS fornecida pelo cliente.

Estrutura ColumnImportance

Uma estrutura que contém o nome e a pontuação de importância de uma coluna.

A importância da coluna ajuda você a entender como as colunas contribuem para o seu modelo, identificando quais delas em seus registros são mais usadas para fazer a correspondência.

Campos

- `ColumnName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome de uma coluna.

- `Importance`: número (double), no máximo 1,0.

  A pontuação de importância da coluna, como um valor decimal.

Operações

- Ação CreateMLTransform (Python: create_ml_transform)
- Ação UpdateMLTransform (Python: update_ml_transform)
- Ação DeleteMLTransform (Python: delete_ml_transform)
- Ação GetMLTransform (Python: get_ml_transform)
- Ação GetMLTransforms (Python: get_ml_transforms)
- Ação ListMLTransforms (Python: list_ml_transforms)
- Ação StartMLEvaluationTaskRun (Python: start_ml_evaluation_task_run)
- Ação StartMLLabelingSetGenerationTaskRun (Python: start_ml_labeling_set_generation_task_run)
- Ação GetMLTaskRun (Python: get_ml_task_run)
- Ação GetMLTaskRuns (Python: get_ml_task_runs)
- Ação CancelMLTaskRun (Python: cancel_ml_task_run)
- Ação StartExportLabelsTaskRun (Python: start_export_labels_task_run)
- Ação StartImportLabelsTaskRun (Python: start_import_labels_task_run)

Ação CreateMLTransform (Python: create_ml_transform)

Cria uma transformação de machine learning do AWS Glue. Essa operação cria a transformação e todos os parâmetros necessários para treiná-la.

Chame esta operação como a primeira etapa no processo de usar uma transformação de machine learning (como a transformação `FindMatches`) para a desduplicação de dados. Você pode fornecer uma `Description`, adicional, além dos parâmetros que você quer usar para seu algoritmo.

Você também deve especificar determinados parâmetros para as tarefas que o AWS Glue executa em seu nome como parte do aprendizado com seus dados e para criação de uma transformação de machine learning de alta qualidade. Esses parâmetros incluem `Role` e, opcionalmente, `AllocatedCapacity`, `Timeout` e `MaxRetries`. Para obter mais informações, consulte Tarefas.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome exclusivo que você atribui à transformação quando a cria.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição da transformação de machine learning que está sendo definida. O padrão é uma string vazia.

- `InputRecordTables` - obrigatório: uma matriz de objetos GlueTable, não mais que dez estruturas.

  Uma lista de definições de tabela do AWS Glue usadas pela transformação.

- `Parameters` – Obrigatório: um objeto TransformParameters.

  Os parâmetros do algoritmo específicos para o tipo de transformação utilizado. Dependente condicionalmente do tipo de transformação.

- `Role` – Obrigatório: string UTF-8.

  O nome ou o nome de recurso da Amazon (ARN) da função do IAM com as permissões necessárias. As permissões necessárias incluem as permissões de função de serviço do AWS Glue para recursos do AWS Glue e as permissões do Amazon S3 exigidas pela transformação.

  - Essa função precisa de permissões de função de serviço do AWS Glue para conceder o acesso aos recursos no AWS Glue. Consulte Anexar uma política aos usuários do IAM que acessam o AWS Glue.
  - Essa função precisa de permissão para suas origens, seus destinos, seu diretório temporário, seus scripts e quaisquer bibliotecas do Amazon Simple Storage Service (Amazon S3) usadas pela execução de tarefa nessa transformação.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

Este valor determina com qual versão de AWS Glue essa transformação de machine learning é compatível. O Glue 1.0 é recomendado para a maioria dos clientes. Se o valor não for definido, a compatibilidade com o Glue 0.9 será definida como padrão. Para obter mais informações, consulte Versões do AWS Glue no guia do desenvolvedor.

- `MaxCapacity` – Número (duplo).

  O número de unidades de processamento de dados (DPUs) do AWS Glue alocadas para execuções de tarefas para essa transformação. Você pode alocar de 2 a 100 DPUs. O padrão é 10. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

  `MaxCapacity` é uma opção mutuamente exclusiva com `NumberOfWorkers` e `WorkerType`.

  - Se `NumberOfWorkers` ou `WorkerType` estiver definido, `MaxCapacity` não poderá ser definido.
  - Se `MaxCapacity` estiver definido, nem `NumberOfWorkers` e nem `WorkerType` poderá ser definido.
  - Se `WorkerType` estiver definido, `NumberOfWorkers` será necessário (e vice-versa).
  - `MaxCapacity` e `NumberOfWorkers` devem ser pelo menos 1.

  Quando o campo `WorkerType` é definido como um valor diferente de `Standard`, o campo `MaxCapacity` é definido automaticamente e se torna somente leitura.

  Quando o campo `WorkerType` é definido como um valor diferente de `Standard`, o campo `MaxCapacity` é definido automaticamente e se torna somente leitura.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido que é alocado quando uma tarefa é executada. Aceita um valor de Standard (Padrão), G.1X ou G.2X.

  - Para o tipo de operador `Standard`, cada operador fornece 4 vCPU, 16 GB de memória e um disco de 50 GB e 2 executores por operador.
  - Para o tipo de operador `G.1X`, cada operador fornece 4 vCPU, 16 GB de memória e um disco e 64 GB, 1 executor por operador.
  - Para o tipo de operador `G.2X`, cada operador fornece 8 vCPU, 32 GB de memória e um disco de 128 GB e 1 executor por operador.

`MaxCapacity` é uma opção mutuamente exclusiva com `NumberOfWorkers` e `WorkerType`.

- Se `NumberOfWorkers` ou `WorkerType` estiver definido, `MaxCapacity` não poderá ser definido.
- Se `MaxCapacity` estiver definido, nem `NumberOfWorkers` e nem `WorkerType` poderá ser definido.
- Se `WorkerType` estiver definido, `NumberOfWorkers` será necessário (e vice-versa).
- `MaxCapacity` e `NumberOfWorkers` devem ser pelo menos 1.

- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de determinado `workerType` que são alocados quando uma tarefa é executada.

  Se `WorkerType` estiver definido, `NumberOfWorkers` será necessário (e vice-versa).

- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite da execução de tarefa para essa transformação, em minutos. Esse é o tempo máximo durante o qual uma execução de tarefa para essa transformação pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas).

- `MaxRetries` – Número (íntegro).

  O número máximo de vezes de novas tentativas dessa transformação após uma execução de tarefa falhar.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags a serem usadas com essa transformação de machine learning. É possível usar tags para limitar o acesso à transformação de machine learning. Para obter mais informações sobre tags no AWS Glue, consulte Tags da AWS no AWS Glue no guia do desenvolvedor.

- `TransformEncryption` – Um objeto TransformEncryption.

  As configurações de criptografia em repouso da transformação que se aplicam ao acesso aos dados do usuário. As transformações de machine learning podem acessar dados do usuário criptografados no Amazon S3 usando o KMS.

Resposta

- `TransformId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um identificador exclusivo gerado para a transformação.

Erros

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `AccessDeniedException`
- `ResourceNumberLimitExceededException`
- `IdempotentParameterMismatchException`

Ação UpdateMLTransform (Python: update_ml_transform)

Atualiza uma transformação de machine learning existente. Chame esta operação para ajustar os parâmetros do algoritmo para atingir resultados melhores.

Depois de chamar essa operação, você pode chamar a operação `StartMLEvaluationTaskRun` para avaliar se os novos parâmetros alcançaram seus objetivos (como melhoria da qualidade da sua transformação de machine learning ou geração de um custo mais eficiente).

Solicitação

- `TransformId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um identificador exclusivo gerado quando a transformação foi criada.
- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome exclusivo que você atribuiu à transformação quando a criou.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição da transformação. O padrão é uma string vazia.

- `Parameters` – Um objeto TransformParameters.

  Os parâmetros de configuração específicos para o tipo de transformação (algoritmo) usado. Dependente condicionalmente do tipo de transformação.

- `Role` – String UTF-8.

  O nome ou o nome de recurso da Amazon (ARN) da função do IAM com as permissões necessárias.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  Este valor determina com qual versão de AWS Glue essa transformação de machine learning é compatível. O Glue 1.0 é recomendado para a maioria dos clientes. Se o valor não for definido, a compatibilidade com o Glue 0.9 será definida como padrão. Para obter mais informações, consulte Versões do AWS Glue no guia do desenvolvedor.

- `MaxCapacity` – Número (duplo).

  O número de unidades de processamento de dados (DPUs) do AWS Glue alocadas para execuções de tarefas para essa transformação. Você pode alocar de 2 a 100 DPUs. O padrão é 10. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

  Quando o campo `WorkerType` é definido como um valor diferente de `Standard`, o campo `MaxCapacity` é definido automaticamente e se torna somente leitura.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido que é alocado quando uma tarefa é executada. Aceita um valor de Standard (Padrão), G.1X ou G.2X.

  - Para o tipo de operador `Standard`, cada operador fornece 4 vCPU, 16 GB de memória e um disco de 50 GB e 2 executores por operador.

  - Para o tipo de operador `G.1X`, cada operador fornece 4 vCPU, 16 GB de memória e um disco e 64 GB, 1 executor por operador.
  - Para o tipo de operador `G.2X`, cada operador fornece 8 vCPU, 32 GB de memória e um disco de 128 GB e 1 executor por operador.
- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de determinado `workerType` que são alocados quando uma tarefa é executada.
- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite para uma tarefa executar essa transformação em minutos. Esse é o tempo máximo durante o qual uma execução de tarefa para essa transformação pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas).
- `MaxRetries` – Número (íntegro).

  O número máximo de vezes de novas tentativas dessa transformação após uma execução de tarefa falhar.

Resposta

- `TransformId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo para a transformação que foi atualizada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `AccessDeniedException`

Ação DeleteMLTransform (Python: delete_ml_transform)

Exclui uma transformação de machine learning do AWS Glue. As transformações de machine learning são um tipo especial de transformação que usa machine learning para aprender os detalhes da transformação que será realizada pelo aprendizado com exemplos fornecidos por humanos. Essas transformações são então salvas pelo AWS Glue. Se você não precisar mais de uma transformação, pode excluí-la chamando `DeleteMLTransforms`. No entanto, qualquer trabalho do AWS Glue que ainda faça referência à transformação excluída não terá mais êxito.

Solicitação

- `TransformId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação que será excluída.

Resposta

- `TransformId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação que foi excluída.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Ação GetMLTransform (Python: get_ml_transform)

Obtém um artefato de transformação de machine learning do AWS Glue e todos os seus metadados correspondentes. As transformações de machine learning são um tipo especial de transformação que usa machine learning para aprender os detalhes da transformação que será realizada pelo aprendizado com exemplos fornecidos por humanos. Essas transformações são então salvas pelo AWS Glue. Você pode recuperar os metadados chamando `GetMLTransform`.

Solicitação

- `TransformId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação, gerado no momento em que a transformação é criada.

Resposta

- `TransformId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação, gerado no momento em que a transformação é criada.
- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome exclusivo atribuído à transformação quando ela foi criada.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição da transformação.
- `Status` – String UTF-8 (valores válidos: `NOT_READY` | `READY` | `DELETING`).

  O último status conhecido da transformação (para indicar se pode ser usada ou não). Um dos "NOT_READY", "READY" ou "DELETING".
- `CreatedOn` – Timestamp.

  A data e a hora em que a transformação foi criada.
- `LastModifiedOn` – Timestamp.

  A data e a hora em que a transformação foi modificada pela última vez.
- `InputRecordTables`: uma matriz de objetos GlueTable, não mais do que dez estruturas.

  Uma lista de definições de tabela do AWS Glue usadas pela transformação.
- `Parameters` – Um objeto TransformParameters.

  Os parâmetros de configuração específicos para o algoritmo usado.
- `EvaluationMetrics` – Um objeto EvaluationMetrics.

As métricas de avaliação mais recentes.

- `LabelCount` – Número (íntegro).

  O número de rótulos disponíveis para essa transformação.

- `Schema`: uma matriz de objetos SchemaColumn, não mais de 100 estruturas.

  O objeto `Map<Column, Type>` que representa o esquema aceito por essa transformação. Tem um limite superior de 100 colunas.

- `Role` – String UTF-8.

  O nome ou o nome de recurso da Amazon (ARN) da função do IAM com as permissões necessárias.

- `GlueVersion` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Custom string pattern #20.

  Este valor determina com qual versão de AWS Glue essa transformação de machine learning é compatível. O Glue 1.0 é recomendado para a maioria dos clientes. Se o valor não for definido, a compatibilidade com o Glue 0.9 será definida como padrão. Para obter mais informações, consulte Versões do AWS Glue no guia do desenvolvedor.

- `MaxCapacity` – Número (duplo).

  O número de unidades de processamento de dados (DPUs) do AWS Glue alocadas para execuções de tarefas para essa transformação. Você pode alocar de 2 a 100 DPUs. O padrão é 10. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória. Para obter mais informações, consulte a página de definição de preços do AWS Glue.

  Quando o campo `WorkerType` é definido como um valor diferente de `Standard`, o campo `MaxCapacity` é definido automaticamente e se torna somente leitura.

- `WorkerType` – String UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  O tipo de operador predefinido que é alocado quando uma tarefa é executada. Aceita um valor de Standard (Padrão), G.1X ou G.2X.

  - Para o tipo de operador `Standard`, cada operador fornece 4 vCPU, 16 GB de memória e um disco de 50 GB e 2 executores por operador.

  - Para o tipo de operador `G.1X`, cada operador fornece 4 vCPU, 16 GB de memória e um disco e 64 GB, 1 executor por operador.
  - Para o tipo de operador `G.2X`, cada operador fornece 8 vCPU, 32 GB de memória e um disco de 128 GB e 1 executor por operador.
- `NumberOfWorkers` – Número (íntegro).

  O número de operadores de determinado `workerType` que são alocados quando uma tarefa é executada.
- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite para uma tarefa executar essa transformação em minutos. Esse é o tempo máximo durante o qual uma execução de tarefa para essa transformação pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas).
- `MaxRetries` – Número (íntegro).

  O número máximo de vezes de novas tentativas dessa transformação após uma execução de tarefa falhar.
- `TransformEncryption` – Um objeto TransformEncryption.

  As configurações de criptografia em repouso da transformação que se aplicam ao acesso aos dados do usuário. As transformações de machine learning podem acessar dados do usuário criptografados no Amazon S3 usando o KMS.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Ação GetMLTransforms (Python: get_ml_transforms)

Obtém uma lista filtrável e classificável de transformações de machine learning do AWS Glue existentes. As transformações de machine learning são um tipo especial de transformação que usa machine learning para aprender os detalhes da transformação que será realizada pelo aprendizado

com exemplos fornecidos por humanos. Essas transformações são então salvas pelo AWS Glue e você pode recuperar os metadados delas chamando `GetMLTransforms`.

Solicitação

- `NextToken` – String UTF-8.

  Um token paginado para equilibrar os resultados.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de resultados a serem retornados.
- `Filter` – Um objeto TransformFilterCriteria.

  Os critérios da transformação de filtros
- `Sort` – Um objeto TransformSortCriteria.

  Os critérios de classificação.

Resposta

- `Transforms` – Obrigatório: uma matriz de objetos MLTransform.

  Uma lista de transformações de machine learning.
- `NextToken` – String UTF-8.

  Um token de paginação, se houver mais resultados disponíveis.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Ação ListMLTransforms (Python: list_ml_transforms)

Recupera uma lista classificável e filtrável de transformações de machine learning do AWS Glue existentes nesta conta da AWS ou os recursos com a tag especificada. Essa operação aceita o

campo `Tags` opcional, que pode ser usado como um filtro das respostas, para que recursos com tags possam ser recuperados como um grupo. Se você optar por usar a filtragem por tags, apenas os recursos com as tags serão recuperados.

Solicitação

- `NextToken` – String UTF-8.

  Um token de continuação, se esta for uma solicitação de continuação.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O tamanho máximo de uma lista a ser retornada.
- `Filter` – Um objeto TransformFilterCriteria.

  Os `TransformFilterCriteria` usados para filtrar as transformações de machine learning.
- `Sort` – Um objeto TransformSortCriteria.

  Os `TransformSortCriteria` usados para classificar as transformações de machine learning.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais do que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  Especifica apenas o retorno desses recursos com tags.

Resposta

- `TransformIds`: obrigatório: uma matriz de strings UTF-8.

  Os identificadores de todas as transformações de machine learning na conta, ou as transformações de machine learning com as tags especificadas.
- `NextToken` – String UTF-8.

  Um token de continuação, se a lista retornada não contiver a métrica mais recente disponível.

Erros

- `EntityNotFoundException`

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Ação StartMLEvaluationTaskRun (Python: start_ml_evaluation_task_run)

Inicia uma tarefa para estimar a qualidade da transformação.

Quando você fornece conjuntos de rótulos como exemplos de verdade, o machine learning do AWS Glue usa esses exemplos para aprender com eles. Os outros rótulos são usados como um teste para estimar a qualidade.

Retorna o identificador exclusivo da execução. Você pode chamar `GetMLTaskRun` para obter mais informações sobre as estatísticas de `EvaluationTaskRun`.

Solicitação

- `TransformId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação de machine learning.

Resposta

- `TaskRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo associado a essa execução.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ConcurrentRunsExceededException`
- `MLTransformNotReadyException`

Ação StartMLLabelingSetGenerationTaskRun (Python: start_ml_labeling_set_generation_task_run)

Inicia o fluxo de trabalho de aprendizado ativo da sua transformação de machine learning para melhorar a qualidade da transformação gerando conjuntos de dados e adicionando rótulos.

Quando `StartMLLabelingSetGenerationTaskRun` estiver concluído, o AWS Glue terá gerado um “conjunto de rótulos” ou um conjunto de questões para serem respondidas por humanos.

No caso da transformação `FindMatches`, essas perguntas são do tipo: "Qual é a maneira correta para agrupar essas linhas em grupos compostos inteiramente por registros correspondentes?"

Depois que o processo de rotulação finalizar, você poderá fazer upload dos seus rótulos com uma chamada para `StartImportLabelsTaskRun`. Depois de `StartImportLabelsTaskRun` terminar, todas as execuções futuras da transformação de machine learning usarão os rótulos novos e melhorados para executar uma transformação de maior qualidade.

Solicitação

- `TransformId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação de machine learning.
- `OutputS3Path` – Obrigatório: string UTF-8.

  O caminho do Amazon Simple Storage Service (Amazon S3) de onde você gera o conjunto de rótulos.

Resposta

- `TaskRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da execução associado a essa execução de tarefa.

Erros

- `EntityNotFoundException`
- `InvalidInputException`

- `OperationTimeoutException`
- `InternalServiceException`
- `ConcurrentRunsExceededException`

Ação GetMLTaskRun (Python: get_ml_task_run)

Obtém os detalhes de uma execução de tarefa em uma transformação de machine learning. As execuções de tarefas de machine learning são tarefas assíncronas executadas pelo AWS Glue em seu nome como parte de vários fluxos de trabalho de machine learning. Você pode verificar as estatísticas de qualquer execução de tarefa chamando `GetMLTaskRun` com o `TaskRunID` e o `TransformID` da sua transformação principal.

Solicitação

- `TransformId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação de machine learning.
- `TaskRunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da tarefa executada.

Resposta

- `TransformId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da tarefa executada.
- `TaskRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador de execução único associado a essa execução.
- `Status` – String UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  O status da execução dessa tarefa.

- `LogGroupName` – String UTF-8.

  Os nomes dos grupos de logs associados à execução de tarefa.

- `Properties` – Um objeto TaskRunProperties.

  A lista de propriedades associadas à execução de tarefa.

- `ErrorString` – String UTF-8.

  As strings de erro associadas à execução de tarefa.

- `StartedOn` – Timestamp.

  A data e hora em que essa execução de tarefa foi iniciada.

- `LastModifiedOn` – Timestamp.

  A data e hora em que essa execução de tarefa foi modificada pela última vez.

- `CompletedOn` – Timestamp.

  A data e hora em que essa execução de tarefa foi finalizada.

- `ExecutionTime` – Número (íntegro).

  A quantidade de tempo (em segundos) em que a tarefa executada consumiu recursos.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Ação GetMLTaskRuns (Python: get_ml_task_runs)

Obtém uma lista de execuções para a transformação de machine learning. As execuções de tarefas de machine learning são tarefas assíncronas executadas pelo AWS Glue em seu nome como parte de vários fluxos de trabalho de machine learning. Você pode obter uma lista de execuções de tarefa de machine learning filtrável e classificável chamando `GetMLTaskRuns` com o `TransformID` da transformação principal e outros parâmetros opcionais conforme documentados nesta seção.

Essa operação retorna uma lista com de execuções históricas e deve ser paginada.

Solicitação

- `TransformId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação de machine learning.
- `NextToken` – String UTF-8.

  Um token de paginação dos resultados. O padrão é vazio.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de resultados a serem retornados.
- `Filter` – Um objeto TaskRunFilterCriteria.

  Os critérios de filtro, na estrutura `TaskRunFilterCriteria`, da execução de tarefa.
- `Sort` – Um objeto TaskRunSortCriteria.

  Os critérios de classificação, na estrutura `TaskRunSortCriteria`, da execução de tarefa.

Resposta

- `TaskRuns` – Uma matriz de objetos TaskRun.

  Uma lista de execuções de tarefa que são associadas a transformação.
- `NextToken` – String UTF-8.

  Um token de paginação, se houver mais resultados disponíveis.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Ação CancelMLTaskRun (Python: cancel_ml_task_run)

Cancela (interrompe) a execução de uma tarefa. As execuções de tarefas de machine learning são tarefas assíncronas executadas pelo AWS Glue em seu nome como parte de vários fluxos de trabalho de machine learning. Você pode cancelar uma execução de tarefa de machine learning a qualquer momento chamando `CancelMLTaskRun` com um `TransformID` de transformação principal da execução de tarefa e o `TaskRunId` da execução de tarefa.

Solicitação

- `TransformId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação de machine learning.
- `TaskRunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um identificador exclusivo para a tarefa executada.

Resposta

- `TransformId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação de machine learning.
- `TaskRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da tarefa executada.
- `Status` – String UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  O status dessa execução.

Erros

- `EntityNotFoundException`
- `InvalidInputException`

- `OperationTimeoutException`
- `InternalServiceException`

Ação StartExportLabelsTaskRun (Python: start_export_labels_task_run)

Inicia uma tarefa assíncrona para exportar todos os dados rotulados para uma determinada transformação. Essa tarefa é a única chamada de API relacionada ao rótulo e não faz parte do fluxo de trabalho típico de aprendizado ativo. Normalmente, você usa `StartExportLabelsTaskRun` quando quer trabalhar com todos os rótulos existentes ao mesmo tempo, por exemplo, quando deseja excluir ou alterar rótulos que foram enviados como verdade anteriormente. Esta operação de API aceita o `TransformId` do qual você deseja exportar rótulos e um caminho do Amazon Simple Storage Service (Amazon S3) para onde exportá-los. A operação retorna um `TaskRunId`. Você pode verificar o status da sua tarefa chamando a API `GetMLTaskRun`.

Solicitação

- `TransformId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação de machine learning.
- `OutputS3Path` – Obrigatório: string UTF-8.

  O caminho do Amazon S3 para onde você exportará os rótulos.

Resposta

- `TaskRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da tarefa executada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Ação StartImportLabelsTaskRun (Python: start_import_labels_task_run)

Permite que você forneça rótulos adicionais (exemplos de verdade) para serem usados no ensino da transformação de machine learning e na melhoria da qualidade. Essa operação de API é geralmente usada como parte do fluxo de aprendizagem ativo que começa com a chamada `StartMLLabelingSetGenerationTaskRun` e que, em última análise, resulta em melhorar a qualidade da sua transformação de machine learning.

Quando `StartMLLabelingSetGenerationTaskRun` terminar, o machine learning do AWS Glue terá gerado várias questões para serem respondidas por humanos. (Responder a essas perguntas é muitas vezes chamado de "rotulamento" nos fluxos de machine learning). No caso da transformação `FindMatches`, essas perguntas são do tipo: "Qual é a maneira correta para agrupar essas linhas em grupos compostos inteiramente por registros correspondentes?" Após o término do processo de rotulamento, os usuários carregam suas respostas/rótulos com uma chamada para `StartImportLabelsTaskRun`. Depois que `StartImportLabelsTaskRun` terminar, todas as execuções futuras da transformação de machine learning usarão os rótulos novos e melhorados para executar uma transformação de maior qualidade.

Por padrão, `StartMLLabelingSetGenerationTaskRun` aprende continuamente e combina todos os rótulos do seu upload, a não ser que você defina `Replace` como true. Se você definir `Replace` como true, `StartImportLabelsTaskRun` exclui e esquece todos os rótulos previamente submetidos e aprende somente com o conjunto exato que você carregar. Substituir rótulos pode ser útil se você perceber que fez um upload incorreto de rótulos anteriormente e acredita que eles estão causando um efeito negativo na qualidade da transformação.

Você pode verificar o status da sua execução de tarefa chamando a operação `GetMLTaskRun`.

Solicitação

- `TransformId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da transformação de machine learning.
- `InputS3Path` – Obrigatório: string UTF-8.

  O caminho do Amazon Simple Storage Service (Amazon S3) para onde você importará os rótulos.
- `ReplaceAllLabels` – Booleano.

  Indica se você deve substituir os rótulos existentes.

Resposta

- `TaskRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador exclusivo da tarefa executada.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`

API Data Quality API

A API Data Quality descreve os tipos de dados de qualidade dos dados e inclui a API para criar, excluir ou atualizar conjuntos de dados, execuções e avaliações de qualidade de dados.

Tipos de dados

- DataSource estrutura
- DataQualityRulesetListDetails estrutura
- DataQualityTargetTable estrutura
- DataQualityRulesetEvaluationRunDescription estrutura
- DataQualityRulesetEvaluationRunFilter estrutura
- DataQualityEvaluationRunAdditionalRunOptions estrutura
- DataQualityRuleRecommendationRunDescription estrutura
- DataQualityRuleRecommendationRunFilter estrutura
- DataQualityResult estrutura
- DataQualityAnalyzerResult estrutura
- DataQualityObservation estrutura
- MetricBasedObservation estrutura

- DataQualityMetricValues estrutura
- DataQualityRuleResult estrutura
- DataQualityResultDescription estrutura
- DataQualityResultFilterCriteria estrutura
- DataQualityRulesetFilterCriteria estrutura

DataSource estrutura

Uma fonte de dados (uma AWS Glue tabela) para a qual você deseja resultados de qualidade de dados.

Campos

- `GlueTable` – Obrigatório: um objeto GlueTable.

  Uma AWS Glue mesa.

DataQualityRulesetListDetails estrutura

Descreve um conjunto de regras de qualidade de dados retornado por `GetDataQualityRuleset`.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do conjunto de regras de qualidade de dados.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do conjunto de regras de qualidade de dados.
- `CreatedOn` – Timestamp.

  A data e hora da criação do conjunto de regras de qualidade de dados.
- `LastModifiedOn` – Timestamp.

  A data e hora da modificação do conjunto de regras de qualidade de dados.

- `TargetTable` – Um objeto DataQualityTargetTable.

  Um objeto representando uma AWS Glue tabela.

- `RecommendationRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Quando um conjunto de regras foi criado de execução de uma recomendação, esse ID de execução é gerado para vincular os dois.

- `RuleCount` – Número (íntegro).

  O número de regras no conjunto de regras.

DataQualityTargetTable estrutura

Um objeto representando uma AWS Glue tabela.

Campos

- `TableName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da AWS Glue tabela.

- `DatabaseName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do banco de dados em que a AWS Glue tabela existe.

- `CatalogId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID do catálogo em que a AWS Glue tabela existe.

DataQualityRulesetEvaluationRunDescription estrutura

Descreve o resultado da avaliação de um conjunto de regras de qualidade de dados.

Campos

- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O identificador de execução único associado a essa execução.

- `Status` – String UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  O status dessa execução.

- `StartedOn` – Timestamp.

  A data e a hora de início da execução.

- `DataSource` – Um objeto DataSource.

  A fonte de dados (uma AWS Glue tabela) associada à execução.

DataQualityRulesetEvaluationRunFilter estrutura

Os critérios de filtro.

Campos

- `DataSource` – Obrigatório: um objeto DataSource.

  Filtre com base em uma fonte de dados (uma AWS Glue tabela) associada à execução.

- `StartedBefore` – Timestamp.

  Filtrar os resultados por execuções iniciadas antes desse horário.

- `StartedAfter` – Timestamp.

  Filtrar os resultados por execuções iniciadas depois desse horário.

DataQualityEvaluationRunAdditionalRunOptions estrutura

Opções adicionais de execução que você pode especificar para uma execução de avaliação.

Campos

- `CloudWatchMetricsEnabled` – Booleano.

  Ativar ou não CloudWatch as métricas.

- `ResultsS3Prefix` – String UTF-8.

Prefixo para o Amazon S3 para armazenar resultados.

- `CompositeRuleEvaluationMethod`: string UTF-8 (valores válidos: `COLUMN` | `ROW`).

  Defina o método de avaliação para regras compostas no conjunto de regras como LINHA/COLUNA

DataQualityRuleRecommendationRunDescription estrutura

Descreve o resultado da avaliação de uma recomendação de regra de qualidade de dados.

Campos

- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador de execução único associado a essa execução.

- `Status` – String UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  O status dessa execução.

- `StartedOn` – Timestamp.

  A data e hora em que essa execução foi iniciada.

- `DataSource` – Um objeto DataSource.

  A fonte de dados (AWS Glue tabela) associada à execução da recomendação.

DataQualityRuleRecommendationRunFilter estrutura

Um filtro para listar as execuções de recomendação de qualidade de dados.

Campos

- `DataSource` – Obrigatório: um objeto DataSource.

  Filtrar com base em uma fonte de dados especificada (AWS Glue tabela).

- `StartedBefore` – Timestamp.

Filtrar com base na hora dos resultados iniciados antes do horário fornecido.

- `StartedAfter` – Timestamp.

  Filtrar com base na hora dos resultados iniciados depois do horário fornecido.

DataQualityResult estrutura

Descreve um resultado de qualidade de dados.

Campos

- `ResultId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um ID de resultado exclusivo para o resultado de qualidade dos dados.

- `Score`: número (double), no máximo 1,0.

  Uma pontuação de qualidade de dados agregada. Representa a razão de regras que foram aproadas para o número total de regras.

- `DataSource` – Um objeto DataSource.

  A tabela associada ao resultado de qualidade dos dados, se houver.

- `RulesetName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do conjunto de regras associado ao resultado de qualidade de dados.

- `EvaluationContext` – String UTF-8.

  No contexto de um trabalho no AWS Glue Studio, cada nó na tela normalmente recebe algum tipo de nome e os nós de qualidade de dados terão nomes. No caso de vários nós, o `evaluationContext` pode diferenciar os nós.

- `StartedOn` – Timestamp.

  A data e hora de início dessa execução de qualidade de dados.

- `CompletedOn` – Timestamp.

  A data e hora de conclusão dessa execução de qualidade de dados.

- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do trabalho associado ao resultado de qualidade dos dados, se houver.
- `JobRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de execução de trabalho associado ao resultado de qualidade dos dados, se houver.
- `RulesetEvaluationRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de execução exclusivo para a avaliação do conjunto de regras para esse resultado de qualidade de dados.
- `RuleResults`: uma matriz de objetos DataQualityRuleResult, não mais de 2.000 estruturas.

  Uma lista de objetos do `DataQualityRuleResult` representando os resultados de cada regra.
- `AnalyzerResults`: uma matriz de objetos DataQualityAnalyzerResult, não mais de 2.000 estruturas.

  Uma lista de objetos `DataQualityAnalyzerResult` representando os resultados de cada analisador.
- `Observations` – Uma matriz de DataQualityObservation objetos, não mais de 50 estruturas.

  Uma lista de objetos `DataQualityObservation` representando as observações geradas após a avaliação das regras e dos analisadores.

DataQualityAnalyzerResult estrutura

Descreve o resultado da avaliação de um analisador de qualidade de dados.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do analisador de qualidade de dados.
- `Description` – String UTF-8 com comprimento não superior a 2048 bytes, correspondente a URI address multi-line string pattern.

Uma descrição do analisador de qualidade de dados.

- `EvaluationMessage` – String UTF-8 com comprimento não superior a 2048 bytes, correspondente a URI address multi-line string pattern.

  Uma mensagem de avaliação.

- `EvaluatedMetrics` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é um número (duplo).

  Um mapa das métricas associadas à avaliação do analisador.

DataQualityObservation estrutura

Descreve a observação gerada após a avaliação das regras e dos analisadores.

Campos

- `Description` – String UTF-8 com comprimento não superior a 2048 bytes, correspondente a URI address multi-line string pattern.

  Uma descrição da observação de qualidade de dados.

- `MetricBasedObservation` – Um objeto MetricBasedObservation.

  Um objeto do tipo `MetricBasedObservation` que representa a observação com base nas métricas de qualidade de dados avaliadas.

MetricBasedObservation estrutura

Descreve a observação baseada em métricas gerada com base nas métricas de qualidade de dados avaliadas.

Campos

- `MetricName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome da métrica de qualidade de dados usada para gerar a observação.

- `MetricValues` – Um objeto DataQualityMetricValues.

  Um objeto do tipo `DataQualityMetricValues` que representa a análise do valor da métrica de qualidade de dados.

- `NewRules` – Uma matriz de strings UTF-8.

  Uma lista de novas regras de qualidade de dados geradas como parte da observação com base no valor da métrica de qualidade de dados.

DataQualityMetricValues estrutura

Descreve o valor da métrica de qualidade de dados de acordo com a análise de dados históricos.

Campos

- `ActualValue` – Número (duplo).

  O valor real da métrica de qualidade de dados.

- `ExpectedValue` – Número (duplo).

  O valor esperado da métrica de qualidade de dados de acordo com a análise de dados históricos.

- `LowerLimit` – Número (duplo).

  O limite inferior do valor da métrica de qualidade de dados de acordo com a análise de dados históricos.

- `UpperLimit` – Número (duplo).

  O limite superior do valor da métrica de qualidade de dados de acordo com a análise de dados históricos.

DataQualityRuleResult estrutura

Descreve o resultado da avaliação de uma regra de qualidade de dados.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da regra de qualidade de dados.

- `Description` – String UTF-8 com comprimento não superior a 2048 bytes, correspondente a URI address multi-line string pattern.

  Uma descrição da regras de qualidade de dados.

- `EvaluationMessage` – String UTF-8 com comprimento não superior a 2048 bytes, correspondente a URI address multi-line string pattern.

  Uma mensagem de avaliação.

- `Result` – String UTF-8 (valores válidos: `PASS` | `FAIL` | `ERROR`).

  Um status de aprovação ou reprovação da regra.

- `EvaluatedMetrics` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é um número (duplo).

  Um mapa das métricas associadas à avaliação da regra.

DataQualityResultDescription estrutura

Descreve um resultado de qualidade de dados.

Campos

- `ResultId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de resultado exclusivo para esse resultado de qualidade dos dados.

- `DataSource` – Um objeto DataSource.

  O nome da tabela associada ao resultado de qualidade dos dados.

- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do trabalho associado ao resultado de qualidade dos dados.

- `JobRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de execução de trabalho associado ao resultado de qualidade dos dados.

- `StartedOn` – Timestamp.

  A hora de início da execução desse resultado de qualidade de dados.

DataQualityResultFilterCriteria estrutura

Critérios usados para retornar resultados de qualidade de dados.

Campos

- `DataSource` – Um objeto DataSource.

  Filtrar os resultados pela fonte de dados especificada. Por exemplo, recuperar todos os resultados de uma AWS Glue tabela.

- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Filtrar os resultados pelo nome de trabalho especificado.

- `JobRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Filtrar os resultados pelo ID e execução de trabalho especificado.

- `StartedAfter` – Timestamp.

  Filtrar os resultados por execuções iniciadas depois desse horário.

- `StartedBefore` – Timestamp.

  Filtrar os resultados por execuções iniciadas antes desse horário.

DataQualityRulesetFilterCriteria estrutura

Os critérios usados para filtrar conjuntos de regras de qualidade de dados.

Campos

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome dos critérios de filtro do conjunto de regras.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  A descrição dos critérios de filtro do conjunto de regras.
- `CreatedBefore` – Timestamp.

  Filtrar por conjuntos de regras criados antes dessa data.
- `CreatedAfter` – Timestamp.

  Filtrar por conjuntos de regras criados depois dessa data.
- `LastModifiedBefore` – Timestamp.

  Filtrar por conjuntos de dados modificados antes dessa data.
- `LastModifiedAfter` – Timestamp.

  Filtrar por conjuntos de dados modificados depois dessa data.
- `TargetTable` – Um objeto DataQualityTargetTable.

  O nome e o nome do banco de dados da tabela de destino.

Operações

- StartDataQualityRulesetEvaluationRun ação (Python: start_data_quality_ruleset_evaluation_run)
- CancelDataQualityRulesetEvaluationRun ação (Python: cancel_data_quality_ruleset_evaluation_run)
- GetDataQualityRulesetEvaluationRun ação (Python: get_data_quality_ruleset_evaluation_run)
- ListDataQualityRulesetEvaluationRuns ação (Python: list_data_quality_ruleset_evaluation_runs)

- StartDataQualityRuleRecommendationRun ação (Python: start_data_quality_rule_recommendation_run)
- CancelDataQualityRuleRecommendationRun ação (Python: cancel_data_quality_rule_recommendation_run)
- GetDataQualityRuleRecommendationRun ação (Python: get_data_quality_rule_recommendation_run)
- ListDataQualityRuleRecommendationRuns ação (Python: list_data_quality_rule_recommendation_runs)
- GetDataQualityResult ação (Python: get_data_quality_result)
- BatchGetDataQualityResult ação (Python: batch_get_data_quality_result)
- ListDataQualityResults ação (Python: list_data_quality_results)
- CreateDataQualityRuleset ação (Python: create_data_quality_ruleset)
- DeleteDataQualityRuleset ação (Python: delete_data_quality_ruleset)
- GetDataQualityRuleset ação (Python: get_data_quality_ruleset)
- ListDataQualityRulesets ação (Python: list_data_quality_rulesets)
- UpdateDataQualityRuleset ação (Python: update_data_quality_ruleset)

StartDataQualityRulesetEvaluationRun ação (Python: start_data_quality_ruleset_evaluation_run)

Depois de ter uma definição de conjunto de regras (recomendada ou própria), você chama essa operação para avaliar o conjunto de regras em relação a uma fonte de dados (tabela).AWS Glue A avaliação calcula os resultados que você pode recuperar com a API `GetDataQualityResult`.

Solicitação

- `DataSource` – Obrigatório: um objeto DataSource.

  A fonte de dados (AWS Glue tabela) associada a essa execução.
- `Role` – Obrigatório: string UTF-8.

  Uma IAM função fornecida para criptografar os resultados da execução.
- `NumberOfWorkers` – Número (íntegro).

  O número de processadores do `G.1X` a serem usados na execução. O padrão é 5.

- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite em minutos para uma execução. Esse é o tempo máximo durante o qual uma execução pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas).

- `ClientToken` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Usado para idempotência e é recomendado que seja definido como um ID aleatório (como um UUID) para evitar criar ou iniciar várias instâncias do mesmo recurso.

- `AdditionalRunOptions` – Um objeto DataQualityEvaluationRunAdditionalRunOptions.

  Opções adicionais de execução que você pode especificar para uma execução de avaliação.

- `RulesetNames` - obrigatório: uma matriz de strings UTF-8, não menos que 1 ou mais que 10 strings.

  Uma lista de nomes de conjuntos de regras.

- `AdditionalDataSources` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é um objeto DataSource A.

  Um mapa de strings de referência para fontes de dados adicionais que você pode especificar para uma execução de avaliação.

Resposta

- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador de execução único associado a essa execução.

Erros

- `InvalidInputException`
- `EntityNotFoundException`

- `OperationTimeoutException`
- `InternalServiceException`
- `ConflictException`

CancelDataQualityRulesetEvaluationRun ação (Python: cancel_data_quality_ruleset_evaluation_run)

Cancela uma execução em que um conjunto de regras está sendo avaliado em relação a uma fonte de dados.

Solicitação

- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador de execução único associado a essa execução.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityRulesetEvaluationRun ação (Python: get_data_quality_ruleset_evaluation_run)

Cancela uma execução específica em que um conjunto de regras está sendo avaliado em relação a uma fonte de dados.

Solicitação

- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador de execução único associado a essa execução.

Resposta

- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador de execução único associado a essa execução.
- `DataSource` – Um objeto DataSource.

  A fonte de dados (uma AWS Glue tabela) associada a essa execução de avaliação.
- `Role` – String UTF-8.

  Uma IAM função fornecida para criptografar os resultados da execução.
- `NumberOfWorkers` – Número (íntegro).

  O número de processadores do `G.1X` a serem usados na execução. O padrão é 5.
- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite em minutos para uma execução. Esse é o tempo máximo durante o qual uma execução pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas).
- `AdditionalRunOptions` – Um objeto DataQualityEvaluationRunAdditionalRunOptions.

  Opções adicionais de execução que você pode especificar para uma execução de avaliação.
- `Status` – String UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  O status dessa execução.
- `ErrorString` – String UTF-8.

  As strings de erro associadas à execução.
- `StartedOn` – Timestamp.

A data e hora em que essa execução foi iniciada.

- `LastModifiedOn` – Timestamp.

  Um carimbo de data/hora. O último ponto em que essa recomendação de regra de qualidade de dados foi executada foi modificado.

- `CompletedOn` – Timestamp.

  A data e hora de conclusão dessa execução.

- `ExecutionTime` – Número (íntegro).

  A quantidade de tempo (em segundos) durante a qual a execução consumiu recursos.

- `RulesetNames`: uma matriz de strings UTF-8, não menos que 1 ou mais que 10 strings.

  Uma lista de nomes de conjuntos de regras para a execução. Atualmente, este parâmetro pode ter apenas um nome de conjunto de regras.

- `ResultIds`: uma matriz de strings UTF-8, não menos que 1 ou mais que 10 strings.

  Uma lista de IDs de resultado para os resultados de qualidade de dados da execução.

- `AdditionalDataSources` – Um array de mapa dos pares de valor-chave.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 255 bytes, correspondente a Single-line string pattern.

  Cada valor é um objeto DataSource A.

  Um mapa de strings de referência para fontes de dados adicionais que você pode especificar para uma execução de avaliação.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRulesetEvaluationRuns ação (Python: list_data_quality_ruleset_evaluation_runs)

Lista todas as execuções que atendem aos critérios de filtro, em que um conjunto de regras é avaliado em relação a uma fonte de dados.

Solicitação

- `Filter` – Um objeto DataQualityRulesetEvaluationRunFilter.

  Os critérios de filtro.

- `NextToken` – String UTF-8.

  Um token paginado para equilibrar os resultados.

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de resultados a serem retornados.

Resposta

- `Runs` – Uma matriz de objetos DataQualityRulesetEvaluationRunDescription.

  Uma lista de objetos `DataQualityRulesetEvaluationRunDescription` que representam execuções de conjuntos de regras de qualidade de dados.

- `NextToken` – String UTF-8.

  Um token de paginação, se houver mais resultados disponíveis.

Erros

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

StartDataQualityRuleRecommendationRun ação (Python: start_data_quality_rule_recommendation_run)

Inicia uma execução de recomendação que é usada para gerar regras quando você não sabe quais regras escrever. AWS Glue O Data Quality analisa os dados e apresenta recomendações para um possível conjunto de regras. Em seguida, você pode fazer a triagem do conjunto de regras e modificar o conjunto de regras gerado de acordo com sua preferência.

As execuções de recomendação são excluídas automaticamente após 90 dias.

Solicitação

- `DataSource` – Obrigatório: um objeto DataSource.

  A fonte de dados (AWS Glue tabela) associada a essa execução.

- `Role` – Obrigatório: string UTF-8.

  Uma IAM função fornecida para criptografar os resultados da execução.

- `NumberOfWorkers` – Número (íntegro).

  O número de processadores do `G.1X` a serem usados na execução. O padrão é 5.

- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite em minutos para uma execução. Esse é o tempo máximo durante o qual uma execução pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas).

- `CreatedRulesetName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um nome para o conjunto de regras.

- `ClientToken` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Usado para idempotência e é recomendado que seja definido como um ID aleatório (como um UUID) para evitar criar ou iniciar várias instâncias do mesmo recurso.

Resposta

- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador de execução único associado a essa execução.

Erros

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ConflictException`

CancelDataQualityRuleRecommendationRun ação (Python: cancel_data_quality_rule_recommendation_run)

Cancela a execução da recomendação especificada que estava sendo usada para gerar regras.

Solicitação

- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador de execução único associado a essa execução.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityRuleRecommendationRun ação (Python: get_data_quality_rule_recommendation_run)

Obtém a execução da recomendação especificada que estava sendo usada para gerar regras.

Solicitação

- `RunId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador de execução único associado a essa execução.

Resposta

- `RunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O identificador de execução único associado a essa execução.
- `DataSource` – Um objeto DataSource.

  A fonte de dados (uma AWS Glue tabela) associada a essa execução.
- `Role` – String UTF-8.

  Uma IAM função fornecida para criptografar os resultados da execução.
- `NumberOfWorkers` – Número (íntegro).

  O número de processadores do `G.1X` a serem usados na execução. O padrão é 5.
- `Timeout` – Número (inteiro), pelo menos 1.

  O tempo limite em minutos para uma execução. Esse é o tempo máximo durante o qual uma execução pode consumir recursos antes de ser encerrada e entrar no status `TIMEOUT`. O padrão é 2.880 minutos (48 horas).
- `Status` – String UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  O status dessa execução.
- `ErrorString` – String UTF-8.

As strings de erro associadas à execução.

- `StartedOn` – Timestamp.

  A data e hora em que essa execução foi iniciada.

- `LastModifiedOn` – Timestamp.

  Um carimbo de data/hora. O último ponto em que essa recomendação de regra de qualidade de dados foi executada foi modificado.

- `CompletedOn` – Timestamp.

  A data e hora de conclusão dessa execução.

- `ExecutionTime` – Número (íntegro).

  A quantidade de tempo (em segundos) durante a qual a execução consumiu recursos.

- `RecommendedRuleset`: string UTF-8, não menos do que 1 ou mais de 65536 bytes de comprimento.

  Quando uma execução de recomendação de regra inicial é concluída, ela cria um conjunto de regras recomendado. Esse membro tem essas regras no formato Data Quality Definition Language (DQDL).

- `CreatedRulesetName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do conjunto de regras que foi criado pela execução.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRuleRecommendationRuns ação (Python: list_data_quality_rule_recommendation_runs)

Lista as execuções de recomendações que atendem aos critérios do filtro.

Solicitação

- `Filter` – Um objeto DataQualityRuleRecommendationRunFilter.

  Os critérios de filtro.
- `NextToken` – String UTF-8.

  Um token paginado para equilibrar os resultados.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de resultados a serem retornados.

Resposta

- `Runs` – Uma matriz de objetos DataQualityRuleRecommendationRunDescription.

  Uma lista dos objetos `DataQualityRuleRecommendationRunDescription`.
- `NextToken` – String UTF-8.

  Um token de paginação, se houver mais resultados disponíveis.

Erros

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityResult ação (Python: get_data_quality_result)

Recupera o resultado de uma avaliação da regra de qualidade de dados.

Solicitação

- `ResultId` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um ID de resultado exclusivo para o resultado de qualidade dos dados.

Resposta

- `ResultId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um ID de resultado exclusivo para o resultado de qualidade dos dados.
- `Score`: número (double), no máximo 1,0.

  Uma pontuação de qualidade de dados agregada. Representa a razão de regras que foram aproadas para o número total de regras.
- `DataSource` – Um objeto DataSource.

  A tabela associada ao resultado de qualidade dos dados, se houver.
- `RulesetName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do conjunto de regras associado ao resultado de qualidade de dados.
- `EvaluationContext` – String UTF-8.

  No contexto de um trabalho no AWS Glue Studio, cada nó na tela normalmente recebe algum tipo de nome e os nós de qualidade de dados terão nomes. No caso de vários nós, o `evaluationContext` pode diferenciar os nós.
- `StartedOn` – Timestamp.

  A data e hora de início da execução desse resultado de qualidade de dados.
- `CompletedOn` – Timestamp.

  A data e hora de conclusão da execução desse resultado de qualidade de dados.
- `JobName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome do trabalho associado ao resultado de qualidade dos dados, se houver.

- `JobRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de execução de trabalho associado ao resultado de qualidade dos dados, se houver.

- `RulesetEvaluationRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID de execução exclusivo associado à avaliação do conjunto de regras.

- `RuleResults`: uma matriz de objetos DataQualityRuleResult, não mais de 2.000 estruturas.

  Uma lista de objetos do `DataQualityRuleResult` representando os resultados de cada regra.

- `AnalyzerResults`: uma matriz de objetos DataQualityAnalyzerResult, não mais de 2.000 estruturas.

  Uma lista de objetos `DataQualityAnalyzerResult` representando os resultados de cada analisador.

- `Observations` – Uma matriz de DataQualityObservation objetos, não mais de 50 estruturas.

  Uma lista de objetos `DataQualityObservation` representando as observações geradas após a avaliação das regras e dos analisadores.

Erros

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `EntityNotFoundException`

BatchGetDataQualityResult ação (Python: batch_get_data_quality_result)

Recupera uma lista de resultados de qualidade de dados para os IDs de resultados especificados.

Solicitação

- `ResultIds` - obrigatório: uma matriz de strings UTF-8, não menos que 1 ou mais que 100 strings.

Uma lista de IDs de resultado exclusivas para os resultados de qualidade dos dados.

Resposta

- `Results` – Obrigatório: uma matriz de objetos DataQualityResult.

  Uma lista de objetos `DataQualityResult` que representam os resultados de qualidade dos dados.
- `ResultsNotFound`: uma matriz de strings UTF-8, não menos que 1 ou mais que 100 strings.

  Uma lista de IDs de resultados para os quais resultados não foram encontrados.

Erros

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityResults ação (Python: list_data_quality_results)

Retorna todos os resultados de execução de qualidade de dados para sua conta.

Solicitação

- `Filter` – Um objeto DataQualityResultFilterCriteria.

  Os critérios de filtro.
- `NextToken` – String UTF-8.

  Um token paginado para equilibrar os resultados.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de resultados a serem retornados.

Resposta

- `Results` – Obrigatório: uma matriz de objetos DataQualityResultDescription.

Uma lista dos objetos `DataQualityResultDescription`.

- `NextToken` – String UTF-8.

  Um token de paginação, se houver mais resultados disponíveis.

Erros

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

CreateDataQualityRuleset ação (Python: create_data_quality_ruleset)

Cria um conjunto de regras de qualidade de dados com regras DQDL aplicadas a uma tabela especificada. AWS Glue

Você cria o conjunto de regras usando a Data Quality Definition Language (DQDL). Para obter mais informações, consulte o guia do AWS Glue desenvolvedor.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um nome exclusivo para o conjunto de regras de qualidade de dados.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição do conjunto de regras de qualidade de dados.
- `Ruleset` - obrigatório: string UTF-8, não menos que 1 ou mais que 65.536 bytes de comprimento.

  Um conjunto de regras em Data Quality Definition Language (DQDL). Para obter mais informações, consulte o guia do AWS Glue desenvolvedor.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

Uma lista de tags aplicadas ao conjunto de regras de qualidade de dados.

- `TargetTable` – Um objeto DataQualityTargetTable.

  Uma tabela de destino associada ao conjunto de regras de qualidade de dados.

- `RecommendationRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um ID de execução exclusivo para a execução da recomendação.

- `ClientToken` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Usado para idempotência e é recomendado que seja definido como um ID aleatório (como um UUID) para evitar criar ou iniciar várias instâncias do mesmo recurso.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um nome exclusivo para o conjunto de regras de qualidade de dados.

Erros

- `InvalidInputException`
- `AlreadyExistsException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`

DeleteDataQualityRuleset ação (Python: delete_data_quality_ruleset)

Exclui um conjunto de regras de qualidade de dados.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um nome para o conjunto de regras de qualidade de dados.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityRuleset ação (Python: get_data_quality_ruleset)

Retorna um conjunto de regras existente por identificador ou um nome.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do grupo de regras.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do grupo de regras.
- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

Uma descrição da predefinição.

- `Ruleset`: string UTF-8, não menos do que 1 ou mais de 65536 bytes de comprimento.

  Um conjunto de regras em Data Quality Definition Language (DQDL). Para obter mais informações, consulte o guia do AWS Glue desenvolvedor.

- `TargetTable` – Um objeto DataQualityTargetTable.

  O nome e o nome do banco de dados da tabela de destino.

- `CreatedOn` – Timestamp.

  Um carimbo de data/hora. A hora e a data de criação desse conjunto de regras de qualidade de dados.

- `LastModifiedOn` – Timestamp.

  Um carimbo de data/hora. O último ponto em que esse conjunto de regras de qualidade de dados foi modificado.

- `RecommendationRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Quando um conjunto de regras foi criado de execução de uma recomendação, esse ID de execução é gerado para vincular os dois.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRulesets ação (Python: list_data_quality_rulesets)

Retorna uma lista paginada de conjuntos de regras para a lista especificada de tabelas. AWS Glue

Solicitação

- `NextToken` – String UTF-8.

Um token paginado para equilibrar os resultados.

- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de resultados a serem retornados.

- `Filter` – Um objeto DataQualityRulesetFilterCriteria.

  Os critérios de filtro.

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  Uma lista de tags de pares chave-valor.

Resposta

- `Rulesets` – Uma matriz de objetos DataQualityRulesetListDetails.

  Uma lista paginada de conjuntos de regras para a lista especificada de tabelas. AWS Glue

- `NextToken` – String UTF-8.

  Um token de paginação, se houver mais resultados disponíveis.

Erros

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

UpdateDataQualityRuleset ação (Python: update_data_quality_ruleset)

Atualiza o conjunto de regras de qualidade de dados especificado.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do conjunto de regras de qualidade de dados.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição da predefinição.

- `Ruleset`: string UTF-8, não menos do que 1 ou mais de 65536 bytes de comprimento.

  Um conjunto de regras em Data Quality Definition Language (DQDL). Para obter mais informações, consulte o guia do AWS Glue desenvolvedor.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do conjunto de regras de qualidade de dados.

- `Description` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma descrição da predefinição.

- `Ruleset`: string UTF-8, não menos do que 1 ou mais de 65536 bytes de comprimento.

  Um conjunto de regras em Data Quality Definition Language (DQDL). Para obter mais informações, consulte o guia do AWS Glue desenvolvedor.

Erros

- `EntityNotFoundException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InvalidInputException`
- `OperationTimeoutException`

- `InternalServiceException`
- `ResourceNumberLimitExceededException`

API de detecção de dados sigilosos

A API Sensitive data detection descreve as APIs usadas para detectar dados confidenciais nas colunas e linhas de seus dados estruturados.

Tipos de dados

- Estrutura CustomEntityType

Estrutura CustomEntityType

Um objeto que representa um padrão personalizado para detectar dados sigilosos nas colunas e linhas dos dados estruturados.

Campos

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um nome para o padrão personalizado que permite que ele seja recuperado ou excluído posteriormente. Esse nome deve ser exclusivo por conta AWS.
- `RegexString` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Uma string de expressão regular usada para detectar dados sigilosos em um padrão personalizado.
- `ContextWords`: uma matriz de strings UTF-8, não menos que 1 ou mais que 20 strings.

  Uma lista de palavras de contexto. Se nenhuma dessas palavras de contexto for encontrada nas proximidades da expressão regular, os dados não serão detectados como dados sigilosos.

  Se nenhuma palavra de contexto for passada, apenas uma expressão regular será marcada.

Operações

Ação CreateCustomEntityType (Python: create_custom_entity_type)

Cria um padrão personalizado usado para detectar dados sigilosos nas colunas e linhas dos dados estruturados.

Cada padrão personalizado criado especifica uma expressão regular e uma lista opcional de palavras de contexto. Se nenhuma palavra de contexto for passada, apenas uma expressão regular será marcada.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Um nome para o padrão personalizado que permite que ele seja recuperado ou excluído posteriormente. Esse nome deve ser exclusivo por conta AWS.
- `RegexString` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Uma string de expressão regular usada para detectar dados sigilosos em um padrão personalizado.
- `ContextWords`: uma matriz de strings UTF-8, não menos que 1 ou mais que 20 strings.

  Uma lista de palavras de contexto. Se nenhuma dessas palavras de contexto for encontrada nas proximidades da expressão regular, os dados não serão detectados como dados sigilosos.

  Se nenhuma palavra de contexto for passada, apenas uma expressão regular será marcada.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

Uma lista de tags aplicadas ao tipo de entidade personalizada.

Resposta

- Name – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do padrão personalizado que você criou.

Erros

- `AccessDeniedException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

Ação DeleteCustomEntityType (Python: delete_custom_entity_type)

Exclui um padrão personalizado especificando seu nome.

Solicitação

- Name – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do padrão personalizado que você deseja excluir.

Resposta

- Name – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

O nome do padrão personalizado que você excluiu.

Erros

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Ação GetCustomEntityType (Python: get_custom_entity_type)

Recupera os detalhes de um padrão personalizado especificando o nome do mesmo.

Solicitação

- `Name` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do padrão personalizado que você deseja recuperar.

Resposta

- `Name` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome do padrão personalizado que você recuperou.
- `RegexString` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  Uma string de expressão regular usada para detectar dados sigilosos em um padrão personalizado.
- `ContextWords`: uma matriz de strings UTF-8, não menos que 1 ou mais que 20 strings.

Uma lista de palavras de contexto, se especificada quando você criou o padrão personalizado. Se nenhuma dessas palavras de contexto for encontrada nas proximidades da expressão regular, os dados não serão detectados como dados sigilosos.

Erros

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Ação BatchGetCustomEntityTypes (Python: batch_get_custom_entity_types)

Recupera os detalhes dos padrões personalizados especificados por uma lista de nomes.

Solicitação

- `Names`: obrigatório: uma matriz de strings UTF-8, não menos de 1 ou mais de 50 strings.

  Uma lista de nomes dos padrões personalizados que você deseja recuperar.

Resposta

- `CustomEntityTypes` – Uma matriz de objetos CustomEntityType.

  Lista de objetos `CustomEntityType` que representam os padrões personalizados que foram criados.
- `CustomEntityTypesNotFound` - Uma matriz de strings UTF-8, não menos que 1 ou mais que 50 strings.

  Uma lista dos nomes de padrões personalizados que não foram encontrados.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Ação ListCustomEntityTypes (Python: list_custom_entity_types)

Lista todos os padrões personalizados que foram criados.

Solicitação

- `NextToken` – String UTF-8.

  Um token paginado para equilibrar os resultados.
- `MaxResults` – Número (inteiro), superior a 1 ou mais que 1000.

  O número máximo de resultados a serem retornados.
- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  Uma lista de tags de pares chave-valor.

Resposta

- `CustomEntityTypes` – Uma matriz de objetos CustomEntityType.

  Lista de objetos `CustomEntityType` que representam os padrões personalizados.
- `NextToken` – String UTF-8.

  Um token de paginação, se houver mais resultados disponíveis.

Erros

- `InvalidInputException`

- `OperationTimeoutException`
- `InternalServiceException`

Marcação de APIs no AWS Glue

Tipos de dados

- Estrutura Tag

Estrutura Tag

O objeto `Tag` representa um rótulo que pode ser atribuído a um recurso da AWS. Cada tag consiste em uma chave e um valor opcional, ambos definidos por você.

Para obter mais informações sobre tags e como controlar o acesso aos recursos no AWS Glue, consulte Marcações da AWS no AWS Glue) e Especificação de ARNs de recurso no AWS Glue no guia do desenvolvedor.

Campos

- `key` – String UTF-8, superior a 1 e inferior a 128 bytes de comprimento.

  A chave de tags. A chave é necessária ao criar uma tag em um objeto. A chave diferencia maiúsculas de minúsculas e não deve conter o prefixo aws.

- `value` – String UTF-8 com comprimento não superior a 256 bytes.

  O valor da tag. O valor é opcional ao criar uma tag em um objeto. O valor diferencia maiúsculas de minúsculas e não deve conter o prefixo aws.

Operações

- Ação TagResource (Python: tag_resource)
- Ação UntagResource (Python: untag_resource)
- Ação GetTags (Python: get_tags)

Ação TagResource (Python: tag_resource)

Adiciona etiquetas a um recurso. Uma tag é um rótulo que pode ser atribuído a um recurso da AWS. No AWS Glue, apenas determinados recursos podem ser marcados. Para obter informações sobre quais recursos podem ser marcados, consulte Tags da AWS no AWS Glue.

Além das permissões de marcação para chamar APIs relacionadas a tags, você também precisa da permissão `glue:GetConnection` para chamar APIs de marcação em conexões e da permissão `glue:GetDatabase` para chamar APIs de marcação em bancos de dados.

Solicitação

- `ResourceArn`: obrigatório: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O ARN do recurso do AWS Glue ao qual se deseja adicionar as tags. Para obter mais informações sobre ARNs de recurso do AWS Glue, consulte o padrão de string de ARN do AWS Glue.
- `TagsToAdd`: obrigatório: uma matriz de mapa dos pares de chave-valor, no máximo 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags a serem adicionadas a esse recurso.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Ação UntagResource (Python: untag_resource)

Remove etiquetas de um recurso.

Solicitação

- `ResourceArn`: obrigatório: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do recurso do qual remover as tags.
- `TagsToRemove`: obrigatório: uma matriz de strings UTF-8, no máximo 50 strings.

  As tags a serem removidas desse recurso.

Resposta

- Nenhum parâmetro de resposta.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Ação GetTags (Python: get_tags)

Recupera uma lista de tags associadas a um recurso.

Solicitação

- `ResourceArn`: obrigatório: string UTF-8, não menos do que 1 ou superior a 10.240 bytes de comprimento, correspondente a Custom string pattern #22.

  O nome de recurso da Amazon (ARN) do recurso para o qual deseja recuperar as tags.

Resposta

- `Tags`: uma matriz de mapa dos pares de chave-valor, não mais que 50 pares.

  Cada chave é uma string UTF-8, com comprimento entre 1 e 128 bytes.

  Cada valor é uma string UTF-8, inferior a 256 bytes de comprimento.

  As tags solicitadas.

Erros

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Tipos de dados comuns

Tipos de dados comuns descrevem os diversos tipos de dados comuns no AWS Glue.

Estrutura Tag

O `Tag` objeto representa um rótulo que você pode atribuir a um AWS recurso. Cada tag consiste em uma chave e um valor opcional, ambos definidos por você.

Para obter mais informações sobre tags e controlar o acesso aos recursos em AWS Glue, consulte AWS Tags em AWS Glue e Especificando ARNs de AWS Glue recursos no guia do desenvolvedor.

Campos

- `key` – String UTF-8, superior a 1 e inferior a 128 bytes de comprimento.

  A chave de tags. A chave é necessária ao criar uma tag em um objeto. A chave diferencia maiúsculas de minúsculas e não deve conter o prefixo aws.
- `value` – String UTF-8 com comprimento não superior a 256 bytes.

  O valor da tag. O valor é opcional ao criar uma tag em um objeto. O valor diferencia maiúsculas de minúsculas e não deve conter o prefixo aws.

DecimalNumber estrutura

Contém um valor numérico em formato decimal.

Campos

- `UnscaledValue`: obrigatório: blob.

  O valor numérico não escalado.
- `Scale`: obrigatório: número (inteiro).

  A escala que determina a posição do ponto decimal no valor não escalado.

ErrorDetail estrutura

Contém detalhes sobre um erro.

Campos

- `ErrorCode` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O código associado a este erro.
- `ErrorMessage` – String de descrição, inferior a 2048 bytes de comprimento, correspondente a URI address multi-line string pattern.

  Uma mensagem descrevendo o erro.

PropertyPredicate estrutura

Define um predicado de propriedade.

Campos

- `Key`: valor de string não superior a 1.024 bytes de comprimento.

  A chave da propriedade.
- `Value`: valor de string não superior a 1.024 bytes de comprimento.

  O valor da propriedade.

- `Comparator` – String UTF-8 (valores válidos: `EQUALS` | `GREATER_THAN` | `LESS_THAN` | `GREATER_THAN_EQUALS` | `LESS_THAN_EQUALS`).

  O comparador costumava comparar esta propriedade com outras.

ResourceUri estrutura

Os URIs para recursos de função.

Campos

- `ResourceType` – String UTF-8 (valores válidos: `JAR` | `FILE` | `ARCHIVE`).

  O tipo de recurso.
- `Uri` – URI (Uniform Resource Identifier), maior que 1 ou maior que 1024 bytes de comprimento, correspondente a URI address multi-line string pattern.

  O URI para acessar o recurso.

ColumnStatistics estrutura

Representa as estatísticas de nível de coluna geradas para uma tabela ou partição.

Campos

- `ColumnName` – Obrigatório: string UTF-8, no mínimo 1 ou mais de 255 bytes de comprimento, correspondente a Single-line string pattern.

  Nome da coluna à qual as estatísticas pertencem.
- `ColumnType`: obrigatório: nome do tipo, no máximo 20.000 bytes de comprimento, correspondente a Single-line string pattern.

  O tipo de dados da coluna.
- `AnalyzedTime`: obrigatório: carimbo de data e hora.

  O carimbo de data e hora do momento em que as estatísticas da coluna foram geradas.
- `StatisticsData` – Obrigatório: um objeto ColumnStatisticsData.

  Um objeto `ColumnStatisticData` que contém os valores de dados de estatísticas.

ColumnStatisticsError estrutura

Encapsula um objeto `ColumnStatistics` que tenha falhado e o motivo da falha.

Campos

- `ColumnStatistics` – Um objeto ColumnStatistics.

  As `ColumnStatistics` da coluna.
- `Error` – Um objeto ErrorDetail.

  Uma mensagem de erro com o motivo da falha de uma operação.

ColumnError estrutura

Encapsula um nome de coluna que tenha falhado e o motivo da falha.

Campos

- `ColumnName` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O nome da coluna que falhou.
- `Error` – Um objeto ErrorDetail.

  Uma mensagem de erro com o motivo da falha de uma operação.

ColumnStatisticsData estrutura

Contém os tipos individuais de dados de estatísticas de coluna. Apenas um objeto de dados deve ser definido e indicado pelo atributo `Type`.

Campos

- `Type`: obrigatório: string UTF-8 (valores válidos: `BOOLEAN` | `DATE` | `DECIMAL` | `DOUBLE` | `LONG` | `STRING` | `BINARY`).

  O tipo de dados de estatísticas de coluna.
- `BooleanColumnStatisticsData` – Um objeto BooleanColumnStatisticsData.

Dados de estatísticas de coluna booleanos.

- `DateColumnStatisticsData` – Um objeto DateColumnStatisticsData.

  Dados de estatísticas de coluna de data.

- `DecimalColumnStatisticsData` – Um objeto DecimalColumnStatisticsData.

  Dados estatísticos da coluna decimal. UnscaledValues dentro estão objetos binários codificados em Base64 que armazenam representações big-endian, complemento de dois, do valor não escalado do decimal.

- `DoubleColumnStatisticsData` – Um objeto DoubleColumnStatisticsData.

  Dados de estatísticas de coluna double.

- `LongColumnStatisticsData` – Um objeto LongColumnStatisticsData.

  Dados de estatísticas de coluna inteiros longos.

- `StringColumnStatisticsData` – Um objeto StringColumnStatisticsData.

  Dados de estatísticas de coluna de string.

- `BinaryColumnStatisticsData` – Um objeto BinaryColumnStatisticsData.

  Dados de estatísticas de coluna binários.

BooleanColumnStatisticsData estrutura

Define estatísticas de coluna suportadas para colunas de dados booleanos.

Campos

- `NumberOfTrues`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores true na coluna.

- `NumberOfFalses`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores false na coluna.

- `NumberOfNulls`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores nulos na coluna.

DateColumnStatisticsData estrutura

Define estatísticas de coluna suportadas para colunas de dados de carimbo de data e hora.

Campos

- `MinimumValue` – Timestamp.

  O valor mais baixo na coluna.
- `MaximumValue` – Timestamp.

  O valor mais alto na coluna.
- `NumberOfNulls`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores nulos na coluna.
- `NumberOfDistinctValues`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores distintos em uma coluna.

DecimalColumnStatisticsData estrutura

Define estatísticas de coluna suportadas para colunas de dados de números de ponto fixo.

Campos

- `MinimumValue` – Um objeto DecimalNumber.

  O valor mais baixo na coluna.
- `MaximumValue` – Um objeto DecimalNumber.

  O valor mais alto na coluna.
- `NumberOfNulls`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores nulos na coluna.
- `NumberOfDistinctValues`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores distintos em uma coluna.

DoubleColumnStatisticsData estrutura

Define estatísticas de coluna suportadas para colunas de dados de números de ponto flutuante.

Campos

- `MinimumValue` – Número (duplo).

  O valor mais baixo na coluna.
- `MaximumValue` – Número (duplo).

  O valor mais alto na coluna.
- `NumberOfNulls`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores nulos na coluna.
- `NumberOfDistinctValues`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores distintos em uma coluna.

LongColumnStatisticsData estrutura

Define estatísticas de coluna suportadas para colunas de dados inteiros.

Campos

- `MinimumValue` – Número (extenso).

  O valor mais baixo na coluna.
- `MaximumValue` – Número (extenso).

  O valor mais alto na coluna.
- `NumberOfNulls`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores nulos na coluna.
- `NumberOfDistinctValues`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores distintos em uma coluna.

StringColumnStatisticsData estrutura

Define estatísticas de coluna suportadas para valores de dados de sequência de caracteres.

Campos

- `MaximumLength`: obrigatório: número (inteiro longo), no máximo nenhum.

  O tamanho da string mais longa na coluna.
- `AverageLength`: obrigatório: número (double), no máximo nenhum.

  O comprimento médio da string na coluna.
- `NumberOfNulls`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores nulos na coluna.
- `NumberOfDistinctValues`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores distintos em uma coluna.

BinaryColumnStatisticsData estrutura

Define estatísticas de coluna suportadas para valores de dados de sequência de bits.

Campos

- `MaximumLength`: obrigatório: número (inteiro longo), no máximo nenhum.

  O tamanho da sequência de bits mais longa na coluna.
- `AverageLength`: obrigatório: número (double), no máximo nenhum.

  O comprimento médio da sequência de bits na coluna.
- `NumberOfNulls`: obrigatório: número (inteiro longo), no máximo nenhum.

  O número de valores nulos na coluna.

Padrões de string

A API usa as seguintes expressões regulares para definir o que é conteúdo válido para vários membros e parâmetros de string:

- Single-line string pattern – "[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\t]*"
- Padrão de string com várias linhas de endereço URI – "[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\r\n\t]*"
- Um padrão de string Logstash Grok – "[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\r\t]*"
- Padrão de string do identificador – "[A-Za-z_][A-Za-z0-9_]*"
- Padrão de string do ARN da AWS IAM – "arn:aws:iam::\d{12}:role/.*"
- Versão do padrão de string – "^[a-zA-Z0-9-_]+$"
- Padrão de string do grupo de logs – "[\.\-_/#A-Za-z0-9]+"
- Padrão de string do stream de logs – "[^:*]*"
- Padrão de string personalizado n.º 10: “[^\r\n]”
- Padrão de string personalizado n.º 11: “^arn:aws(-(cn|us-gov|iso(-[bef])?))?:secretsmanager:.*$”
- Padrão de string personalizado n.º 12: “^(https?)://[-a-zA-Z0-9+@#/%?=~_|!:,.;]*[-a-zA-Z0-9+@#/%=~_|]”
- Padrão de string personalizado n.º 13: “\S+”
- Padrão de string personalizado n.º 14: “^(https?):\/\/[^\s/$.?#].[^\s]*$”
- Padrão de string personalizado n.º 15: “^subnet-[a-z0-9]+$”
- Padrão de string personalizado n.º 16: “[\p{L}\p{N}\p{P}]*”
- Padrão de string personalizado n.º 17: “[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}”
- Padrão de string personalizado n.º 18: “[a-zA-Z0-9-_$#.]+”
- Padrão de string personalizado n.º 19: “^\w+\.\w+\.\w+$”
- Padrão de string personalizado n.º 20: “^\w+\.\w+$”
- Padrão de string personalizado n.º 21: “^([2-3]|3[.]9)$”
- Padrão de string personalizado n.º 22: “arn:(aws|aws-us-gov|aws-cn):glue:.*”
- Padrão de string personalizado n.º 23: "(^arn:aws:iam::\w{12}:root)"
- Padrão de string personalizado n.º 24: "^arn:aws(-(cn|us-gov|iso(-[bef])?))?:iam::[0-9]{12}:role/.+"
- Padrão de string personalizado n.º 25: "arn:aws:kms:.*"

- Padrão de string personalizado n.º 26: `"arn:aws[^:]*:iam::[0-9]*:role/.+"`
- Padrão de string personalizado n.º 27: `"[\.\-_A-Za-z0-9]+"`
- Padrão de string personalizado n.º 28: `"^s3://([^/]+)/([^/]+/)*([^/]+)$"`
- Padrão de string personalizado n.º 29: `".*"`
- Padrão de string personalizado n.º 30: `"^(Sun|Mon|Tue|Wed|Thu|Fri|Sat):([01]?[0-9]|2[0-3])$"`
- Padrão de string personalizado n.º 31: `"[a-zA-Z0-9_.-]+"`
- Padrão de string personalizado n.º 32: `".*\S.*"`
- Padrão de string personalizado n.º 33: `"[a-zA-Z0-9-=._/@]+"`
- Padrão de string personalizado n.º 34: `"[1-9][0-9]*|[1-9][0-9]*-[1-9][0-9]*"`
- Padrão de string personalizado n.º 35: `"[\s\S]*"`
- Padrão de string personalizado n.º 36: `"([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\S\r\n"'= ;])*"`
- Padrão de string personalizado n.º 37: `"[*A-Za-z0-9_-]*"`
- Padrão de string personalizado n.º 38: `"([\u0020-\u007E\r\s\n])*"`
- Padrão de string personalizado n.º 39: `"[A-Za-z0-9_-]*"`
- Padrão de string personalizado n.º 40: `"([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\S\r\n"'])*"`
- Padrão de string personalizado n.º 41: `"([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\S\r\n])*"`
- Padrão de string personalizado n.º 42: `"([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\s])*"`
- Padrão de string personalizado #43 — `"([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\r\n])*”`

Exceções

Esta seção descreve as exceções do AWS Glue que você pode usar para encontrar a origem dos problemas e corrigi-los. Para obter mais informações sobre códigos de erro HTTP e strings para exceções relacionadas a machine learning, consulte the section called “Exceções de machine learning do AWS Glue”.

Estrutura AccessDeniedException

O acesso a um recurso foi negado.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura AlreadyExistsException

Um recurso a ser criado ou adicionado já existe.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura ConcurrentModificationException

Dois processos estão tentando modificar um recurso simultaneamente.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura ConcurrentRunsExceededException

Muitos trabalhos estão sendo executados simultaneamente.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura CrawlerNotRunningException

O crawler especificado não está sendo executado.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura CrawlerRunningException

A operação não pode ser realizada porque o crawler já está sendo executado.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura CrawlerStoppingException

O crawler especificado está sendo interrompido.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura EntityNotFoundException

Uma entidade especificada não existe

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.
- `FromFederationSource` – Booleano.

Indica se a exceção está relacionada ou não a uma fonte federada.

Estrutura FederationSourceException

Uma fonte de federação falhou.

Campos

- `FederationSourceErrorCode`: string UTF-8 (valores válidos: `AccessDeniedException` | `EntityNotFoundException` | `InvalidCredentialsException` | `InvalidInputException` | `InvalidResponseException` | `OperationTimeoutException` | `OperationNotSupportedException` | `InternalServiceException` | `PartialFailureException` | `ThrottlingException`).

  O código de erro do problema.
- `Message` – String UTF-8.

  A mensagem descrevendo o problema.

Estrutura FederationSourceRetryableException

Uma fonte de federação falhou, mas a operação pode ser repetida.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura de GlueEncryptionException

Uma operação de criptografia falhou.

Campos

- `Message` – String UTF-8.

  A mensagem descrevendo o problema.

Estrutura IdempotentParameterMismatchException

O mesmo identificador exclusivo foi associado a dois registros diferentes.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura IllegalWorkflowStateException

O fluxo de trabalho está em um estado inválido para executar uma operação solicitada.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura InternalServiceException

Ocorreu um erro interno de serviço.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura InvalidExecutionEngineException

Um mecanismo de execução desconhecido ou inválido foi especificado.

Campos

- `message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura InvalidInputException

A entrada fornecida não é válida.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.
- `FromFederationSource` – Booleano.

  Indica se a exceção está relacionada ou não a uma fonte federada.

Estrutura InvalidStateException

Um erro que indica que seus dados estão em um estado inválido.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura InvalidTaskStatusTransitionException

A transição adequada de uma tarefa para a próxima falhou.

Campos

- `message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura JobDefinitionErrorException

Uma definição de trabalho não é válida.

Campos

- `message` – String UTF-8.

Uma mensagem descrevendo o problema.

Estrutura JobRunInTerminalStateException

O estado terminal de uma execução de trabalho sinaliza uma falha.

Campos

- `message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura JobRunInvalidStateTransitionException

Uma execução de trabalho encontrou uma transição inválida do estado de origem para o estado de destino.

Campos

- `jobRunId` – String UTF-8, superior a 1 e inferior a 255 bytes de comprimento, correspondente a Single-line string pattern.

  O ID da execução de trabalho em questão.
- `message` – String UTF-8.

  Uma mensagem descrevendo o problema.
- `sourceState`: string UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING` | `EXPIRED`).

  O estado de origem.
- `targetState`: string UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING` | `EXPIRED`).

  O estado do destino.

Estrutura JobRunNotInTerminalStateException

Uma execução de trabalho não está em um estado terminal.

Campos

- `message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura LateRunnerException

Um executor de trabalho está atrasado.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura NoScheduleException

Não há uma programação aplicável.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura OperationTimeoutException

A operação expirou.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura ResourceNotReadyException

Um recurso não estava pronto para uma transação.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura ResourceNumberLimitExceededException

Um limite numérico de recursos foi excedido.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura SchedulerNotRunningException

O programador especificado não está sendo executado.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura SchedulerRunningException

O programador especificado já está sendo executado.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura SchedulerTransitioningException

O programador especificado está em transição.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura UnrecognizedRunnerException

O executor de trabalho não foi reconhecido.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura ValidationException

Não foi possível validar um valor.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

Estrutura VersionMismatchException

Houve um conflito de versão.

Campos

- `Message` – String UTF-8.

  Uma mensagem descrevendo o problema.

AWS Glue Exemplos de código de API usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Glue com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Conceitos básicos

Olá AWS Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte ListJobsa Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMake C MakeLists .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 para o arquivo de origem 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 obter detalhes da API, consulte ListJobsa Referência AWS SDK for C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte ListJobsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte ListJobsa Referência AWS SDK for JavaScript da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte a ListJobsreferência da API AWS SDK for Rust.

Exemplos de código

- Ações para AWS Glue usar AWS SDKs
  - Use CreateCrawler com um AWS SDK ou CLI
  - Use CreateJob com um AWS SDK ou CLI
  - Use DeleteCrawler com um AWS SDK ou CLI
  - Use DeleteDatabase com um AWS SDK ou CLI
  - Use DeleteJob com um AWS SDK ou CLI
  - Use DeleteTable com um AWS SDK ou CLI
  - Use GetCrawler com um AWS SDK ou CLI
  - Use GetDatabase com um AWS SDK ou CLI
  - Use GetDatabases com um AWS SDK ou CLI
  - Use GetJob com um AWS SDK ou CLI
  - Use GetJobRun com um AWS SDK ou CLI
  - Use GetJobRuns com um AWS SDK ou CLI
  - Use GetTables com um AWS SDK ou CLI
  - Use ListJobs com um AWS SDK ou CLI
  - Use StartCrawler com um AWS SDK ou CLI
  - Use StartJobRun com um AWS SDK ou CLI
- Cenários para AWS Glue usar AWS SDKs
  - Comece a executar AWS Glue rastreadores e trabalhos usando um SDK AWS

Ações para AWS Glue usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS Glue ações individuais com AWS SDKs. Esses trechos chamam a AWS Glue API e são trechos de código de programas maiores que devem ser executados em contexto. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Glue.

Exemplos

- Use CreateCrawler com um AWS SDK ou CLI
- Use CreateJob com um AWS SDK ou CLI
- Use DeleteCrawler com um AWS SDK ou CLI
- Use DeleteDatabase com um AWS SDK ou CLI
- Use DeleteJob com um AWS SDK ou CLI
- Use DeleteTable com um AWS SDK ou CLI
- Use GetCrawler com um AWS SDK ou CLI
- Use GetDatabase com um AWS SDK ou CLI
- Use GetDatabases com um AWS SDK ou CLI
- Use GetJob com um AWS SDK ou CLI
- Use GetJobRun com um AWS SDK ou CLI
- Use GetJobRuns com um AWS SDK ou CLI
- Use GetTables com um AWS SDK ou CLI
- Use ListJobs com um AWS SDK ou CLI
- Use StartCrawler com um AWS SDK ou CLI
- Use StartJobRun com um AWS SDK ou CLI

Use **`CreateCrawler`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `CreateCrawler`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
   /// <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 obter detalhes da API, consulte CreateCrawlera Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte CreateCrawlera Referência AWS SDK for C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte CreateCrawlera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte CreateCrawlera Referência AWS SDK for JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a CreateCrawlerreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        $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 obter detalhes da API, consulte CreateCrawlera Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a CreateCrawlerReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte CreateCrawlera Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte a CreateCrawlerreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`CreateJob`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `CreateJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
   /// <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 obter detalhes da API, consulte CreateJoba Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte CreateJoba Referência AWS SDK for C++ da API.

CLI

AWS CLI

Para criar um trabalho para transformar dados

O exemplo de `create-job` a seguir cria um trabalho de streaming que executa um script armazenado no 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
```

Conteúdo 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()
    }
}
```

Saída:

```
{
```

```
    "Name": "my-testing-job"
}
```

Para obter mais informações, consulte Authoring Jobs in AWS Glue no Glue Developer Guide.AWS

- Para obter detalhes da API, consulte CreateJobem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte CreateJoba Referência AWS SDK for JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        $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 obter detalhes da API, consulte CreateJoba Referência AWS SDK for PHP da API.

PowerShell

Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria uma nova tarefa no AWS Glue. O valor do nome do comando é sempre**`glueetl`**. AWS O Glue é compatível com a execução de scripts de tarefas escritos em Python ou Scala. Neste exemplo, o script de trabalho (MyTestGlueJob.py) é escrito em Python. Os parâmetros do Python são especificados na **`$DefArgs`** variável e, em seguida, passados para o PowerShell comando no **`DefaultArguments`** parâmetro, que aceita uma tabela de hash. Os parâmetros na **`$JobParams`** variável vêm da CreateJob API, documentados no tópico Jobs (https://docs.aws.amazon.com/glue/latest/dg/ aws-glue-api-jobs -job.html) da referência da API 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 obter detalhes da API, consulte CreateJobem Referência de AWS Tools for PowerShell cmdlet.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a CreateJobReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte CreateJoba Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte a CreateJobreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteCrawler`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `DeleteCrawler`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
    /// <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 obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK for C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });
```

```
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK for JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a DeleteCrawlerReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a DeleteCrawlerreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteDatabase`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `DeleteDatabase`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
    /// <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 obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK for C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });
```

```
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK for JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a DeleteDatabaseReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a DeleteDatabasereferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteJob`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `DeleteJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
    /// <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 obter detalhes da API, consulte DeleteJoba Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte DeleteJoba Referência AWS SDK for C++ da API.

CLI

AWS CLI

Para excluir um trabalho

O exemplo de `delete-job` a seguir exclui um trabalho que não é mais necessário.

```
aws glue delete-job \
    --job-name my-testing-job
```

Saída:

```
{
    "JobName": "my-testing-job"
}
```

Para obter mais informações, consulte Trabalhando com trabalhos no AWS Glue Console no AWS Glue Developer Guide.

- Para obter detalhes da API, consulte DeleteJobna Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteJoba Referência AWS SDK for JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
```

```
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteJoba Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a DeleteJobReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte DeleteJoba Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a DeleteJobreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteTable`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `DeleteTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
    /// <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 obter detalhes da API, consulte DeleteTablea Referência AWS SDK for .NET da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
```

```
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK for JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte DeleteTablea Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a DeleteTableReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte DeleteTablea Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }
```

- Para obter detalhes da API, consulte a DeleteTablereferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetCrawler`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `GetCrawler`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
    /// <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 obter detalhes da API, consulte GetCrawlera Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
       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 obter detalhes da API, consulte GetCrawlera Referência AWS SDK for C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte GetCrawlera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetCrawlera Referência AWS SDK for JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a GetCrawlerreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte GetCrawlera Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a GetCrawlerReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte GetCrawlera Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
            let tmp_crawler = glue
                .get_crawler()
                .name(self.crawler())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a GetCrawlerreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetDatabase`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `GetDatabase`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
    /// <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 obter detalhes da API, consulte GetDatabasea Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
       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 obter detalhes da API, consulte GetDatabasea Referência AWS SDK for C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte GetDatabasea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetDatabasea Referência AWS SDK for JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a GetDatabasereferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        $databaseName = "doc-example-database-$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 obter detalhes da API, consulte GetDatabasea Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a GetDatabaseReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte GetDatabasea Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte a GetDatabasereferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetDatabases`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `GetDatabases`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

CLI

AWS CLI

Para listar as definições de alguns ou de todos os bancos de dados no AWS Glue Data Catalog

O exemplo de `get-databases` a seguir retorna informações sobre os bancos de dados no Catálogo de Dados.

```
aws glue get-databases
```

Saída:

```
{
    "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 obter mais informações, consulte Definir um banco de dados no seu Catálogo de Dados no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte GetDatabasesem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const getDatabases = () => {
  const client = new GlueClient({});

  const command = new GetDatabasesCommand({});

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetDatabasesa Referência AWS SDK for JavaScript da API.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetJob`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `GetJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

CLI

AWS CLI

Para recuperar informações sobre um trabalho

O exemplo de `get-job` a seguir recupera informações sobre um trabalho.

```
aws glue get-job \
    --job-name my-testing-job
```

Saída:

```
{
    "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 obter mais informações, consulte Trabalhos no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte GetJobem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const getJob = (jobName) => {
  const client = new GlueClient({});

  const command = new GetJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetJoba Referência AWS SDK for JavaScript da API.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetJobRun`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `GetJobRun`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
    /// <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 obter detalhes da API, consulte GetJobRuna Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte GetJobRuna Referência AWS SDK for C++ da API.

CLI

AWS CLI

Para obter informações sobre uma execução de trabalho

O exemplo `get-job-run` a seguir recupera informações sobre uma execução de trabalho.

```
aws glue get-job-run \
    --job-name "Combine legistators data" \
    --run-id jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e
```

Saída:

```
{
```

```
    "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 obter mais informações, consulte Execuções de trabalhos no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte GetJobRunem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetJobRuna Referência AWS SDK for JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        $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 obter detalhes da API, consulte GetJobRuna Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a GetJobRunReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte GetJobRuna Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte a GetJobRunreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetJobRuns`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `GetJobRuns`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
    /// <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 obter detalhes da API, consulte GetJobRunsa Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte GetJobRunsa Referência AWS SDK for C++ da API.

CLI

AWS CLI

Para obter informações sobre todas as execuções de um trabalho

O exemplo `get-job-runs` a seguir recupera informações sobre execuções de trabalho para um trabalho.

```
aws glue get-job-runs \
    --job-name "my-testing-job"
```

Saída:

```
{
    "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 obter mais informações, consulte Execuções de trabalhos no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte GetJobRunsem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetJobRunsa Referência AWS SDK for JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        $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 obter detalhes da API, consulte GetJobRunsa Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a GetJobRunsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte GetJobRunsa Referência AWS SDK for Ruby da API.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetTables`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `GetTables`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
/// <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 obter detalhes da API, consulte GetTablesa Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
       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 obter detalhes da API, consulte GetTablesa Referência AWS SDK for C++ da API.

CLI

AWS CLI

Para listar as definições de algumas ou de todas as tabelas no banco de dados especificado

O exemplo de `get-tables` a seguir retorna informações sobre as tabelas no banco de dados especificado.

```
aws glue get-tables --database-name 'tempdb'
```

Saída:

```
{
    "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 obter mais informações, consulte Definindo tabelas no catálogo de dados do AWS Glue no AWS Glue Developer Guide.

- Para obter detalhes da API, consulte GetTablesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte GetTablesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetTablesa Referência AWS SDK for JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
    {
```

```
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- Para obter detalhes da API, consulte GetTablesa Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a GetTablesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte GetTablesa Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

- Para obter detalhes da API, consulte a GetTablesreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListJobs`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `ListJobs`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
    /// <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 obter detalhes da API, consulte ListJobsa Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte ListJobsa Referência AWS SDK for C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const listJobs = () => {
  const client = new GlueClient({});

  const command = new ListJobsCommand({});

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK for JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        $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 obter detalhes da API, consulte ListJobsa Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a ListJobsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte ListJobsa Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
    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 obter detalhes da API, consulte a ListJobsreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`StartCrawler`** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `StartCrawler`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
/// <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 obter detalhes da API, consulte StartCrawlera Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
       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 obter detalhes da API, consulte StartCrawlera Referência AWS SDK for C++ da API.

CLI

AWS CLI

Para iniciar um crawler

O exemplo de `start-crawler` a seguir inicia um crawler.

```
aws glue start-crawler --name my-crawler
```

Saída:

```
None
```

Para obter mais informações, consulte Definição de crawlers no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte StartCrawlerna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte StartCrawlera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte StartCrawlera Referência AWS SDK for JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a StartCrawlerreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        $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 obter detalhes da API, consulte StartCrawlera Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a StartCrawlerReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte StartCrawlera Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte a StartCrawlerreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **StartJobRun** com um AWS SDK ou CLI

Os exemplos de códigos a seguir mostram como usar `StartJobRun`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- Começar a executar crawlers e trabalhos

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
/// <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 obter detalhes da API, consulte StartJobRuna Referência AWS SDK for .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
       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 obter detalhes da API, consulte StartJobRuna Referência AWS SDK for C++ da API.

CLI

AWS CLI

Para iniciar a execução de um trabalho

O exemplo de `start-job-run` a seguir inicia um trabalho.

```
aws glue start-job-run \
    --job-name my-job
```

Saída:

```
{
    "JobRunId":
 "jr_22208b1f44eb5376a60569d4b21dd20fcb8621e1a366b4e7b2494af764b82ded"
}
```

Para obter mais informações, consulte Criação de trabalhos no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte StartJobRunem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte StartJobRuna Referência AWS SDK for JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        $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 obter detalhes da API, consulte StartJobRuna Referência AWS SDK for PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte a StartJobRunReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
# 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 obter detalhes da API, consulte StartJobRuna Referência AWS SDK for Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
        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 obter detalhes da API, consulte a StartJobRunreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Cenários para AWS Glue usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS Glue com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS Glue. Cada cenário inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código.

Exemplos

- Comece a executar AWS Glue rastreadores e trabalhos usando um SDK AWS

Comece a executar AWS Glue rastreadores e trabalhos usando um SDK AWS

Os exemplos de código a seguir mostram como:

- Criar um crawler que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados formatado em CSV.
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Criar um trabalho para extrair dados em CSV do bucket do S3, transformá-los e carregar a saída formatada em JSON em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

.NET

AWS SDK for .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as AWS Glue funções usadas no cenário.

```
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;
    }

}
```

Crie uma classe que execute o cenário.

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API 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**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API 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**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API 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**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

Crie e execute um crawler que examine um bucket público do Amazon Simple Storage Service (Amazon S3) e gere um banco de dados de metadados que descreva os dados no formato CSV que encontrar.

```
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 };
  };
```

Liste informações sobre bancos de dados e tabelas em seu 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 };
  };
```

Crie e execute um trabalho que extraia dados em CSV do bucket do Amazon S3 de origem, transforme-os removendo e renomeando campos, e carregue a saída formatada em JSON em outro bucket do 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 };
  };
```

Liste informações sobre execuções de tarefas e visualize alguns dos dados 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 };
  };
```

Exclua todos os recursos criados pela demonstração.

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API 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**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API 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**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API 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**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as AWS Glue funções usadas no cenário.

```
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
```

Crie uma classe que execute o cenário.

```
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.")
```

Crie um script ETL que seja usado AWS Glue para extrair, transformar e carregar dados durante a execução do trabalho.

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API 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**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as AWS Glue funções usadas no cenário.

```
# 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
```

Crie uma classe que execute o cenário.

```
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
```

Crie um script ETL que seja usado AWS Glue para extrair, transformar e carregar dados durante a execução do trabalho.

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API 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**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no AWS Code Examples Repository.

Crie e execute um crawler que examine um bucket público do Amazon Simple Storage Service (Amazon S3) e gere um banco de dados de metadados que descreva os dados no formato CSV que encontrar.

```
        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)),
                }
            }
        }?;
```

Liste informações sobre bancos de dados e tabelas em seu 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();
```

Crie e execute um trabalho que extraia dados em CSV do bucket do Amazon S3 de origem, transforme-os removendo e renomeando campos, e carregue a saída formatada em JSON em outro bucket do 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();
```

Exclua todos os recursos criados pela demonstração.

```
        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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Rust.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler

- DeleteDatabase
- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando esse serviço com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Segurança no AWS Glue

A segurança na nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você contará com um datacenter e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem: a AWS é responsável pela proteção da infraestrutura que executa produtos da AWS na Nuvem AWS. A AWS também fornece serviços que podem ser usados com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos compliance programsAWS. Para saber mais sobre os programas de conformidade que se aplicam ao AWS Glue, consulte AWS Services in Scope by Compliance Program.
- Segurança na nuvem: sua responsabilidade é determinada pelo serviço da AWS que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda a entender como aplicar o modelo de responsabilidade compartilhada ao usar o .AWS Glue Os tópicos a seguir mostram como configurar o AWS Glue para atender aos seus objetivos de segurança e conformidade. Saiba como usar outros serviços AWS que ajudam a monitorar e proteger os recursos do AWS Glue.

Tópicos

Proteção de dados no AWS Glue

O AWS Glue oferece vários recursos que foram projetados para ajudar a proteger seus dados.

Tópicos

Criptografia inativa

O AWS Glue oferece suporte à criptografia de dados em repouso para Criar trabalhos ETL visuais com o AWS Glue Studio e Desenvolver scripts com endpoints de desenvolvimento. Você pode configurar trabalhos de extração, transformação e carregamento (ETL) e endpoints de desenvolvimento para usar chaves do AWS Key Management Service (AWS KMS) para gravar dados criptografados em repouso. Também é possível criptografar os metadados armazenados no AWS Glue Data Catalog usando chaves que você gerencia com o AWS KMS. Além disso, você pode usar chaves do AWS KMS para criptografar marcadores de trabalho e os logs gerados pelos crawlers e trabalhos de ETL.

Você pode criptografar objetos de metadados em seu AWS Glue Data Catalog, além dos dados gravados no Amazon Simple Storage Service (Amazon S3) e no Amazon CloudWatch Logs, por trabalhos, crawlers e endpoints de desenvolvimento. Ao criar trabalhos, crawlers e endpoints de desenvolvimentos no AWS Glue, é possível fornecer configurações de criptografia anexando uma configuração de segurança. As configurações de segurança contêm chaves de criptografia no lado do servidor gerenciadas pelo Amazon S3 (SSE-S3) ou chaves mestras do cliente (CMKs) armazenadas no AWS KMS (SSE-KMS). É possível criar configurações de segurança usando o console do AWS Glue.

Também é possível ativar a criptografia de todo o Data Catalog na conta. Faça isso especificando CMKs armazenadas no AWS KMS.

> **Important**
>
> O AWS Glue oferece suporte somente a chaves gerenciadas pelo cliente simétricas. Para obter mais informações, consulte Chaves gerenciadas pelo cliente (CMKs) no Guia do desenvolvedor do AWS Key Management Service.

Com a criptografia ativada, quando você adiciona objetos do Data Catalog, executa crawlers ou trabalhos ou inicia endpoints de desenvolvimento, as chaves SSE-S3 ou SSE-KMS são usadas para gravar dados em repouso. Além disso, você pode configurar AWS Glue para acessar apenas armazenamentos de dados Java Database Connectivity (JDBC) por meio de um protocolo Transport Layer Security (TLS) confiável.

No AWS Glue, você controla as configurações de criptografia nos seguintes locais:

- As configurações do seu Data Catalog.
- As configurações de segurança que você cria.
- A configuração de criptografia no lado do servidor (SSE-S3 ou SSE-KMS) que é transmitida como um parâmetro para o trabalho de ETL (extração, transformação e carregamento) do AWS Glue.

Para obter mais informações sobre como definir a criptografia, consulte Configurar criptografia no AWS Glue.

Tópicos

- Como criptografar seu Data Catalog
- Criptografar senhas de conexão
- Criptografar dados gravados pelo AWS Glue

Como criptografar seu Data Catalog

A criptografia do AWS Glue Data Catalog fornece segurança aprimorada para seus dados confidenciais. O AWS Glue se integra ao AWS Key Management Service (AWS KMS) para criptografar metadados armazenados no Catálogo de Dados. Você pode habilitar ou desabilitar as configurações de criptografia para recursos no Catálogo de Dados usando o console do AWS Glue ou a AWS CLI.

Quando você habilita a criptografia para seu Catálogo de Dados, todos os novos objetos que você criar serão criptografados. Quando você desabilita a criptografia, os novos objetos criados não serão criptografados, mas os objetos criptografados existentes permanecerão criptografados.

Você pode criptografar todo o seu Catálogo de Dados usando chaves de criptografia gerenciadas pela AWS ou chaves de criptografia gerenciadas pelo cliente. Para obter mais informações sobre os principais tipos e estados, consulte Conceitos do AWS Key Management Service no Guia do desenvolvedor do AWS Key Management Service.

Chaves gerenciadas pela AWS

Chaves gerenciadas pelo AWS são chaves do KMS em sua conta que são criadas, gerenciadas e usadas em seu nome por um serviço da AWS integrado ao AWS KMS. É possível visualizar as chaves gerenciadas pela AWS na conta, visualize suas políticas de chaves e audite seu uso nos logs do AWS CloudTrail. No entanto, não é possível gerenciar essas chaves nem alterar suas permissões.

A criptografia em repouso se integra automaticamente com o AWS KMS para gerenciar as chaves gerenciadas pela AWS para o AWS Glue que são usadas para criptografar seus metadados. Se uma chave gerenciada pela AWS não existir quando a criptografia de metadados for habilitada, o AWS KMS criará automaticamente uma nova chave para você.

Para obter mais informações, consulte Chaves gerenciadas pela AWS.

Chaves gerenciadas pelo cliente

Chaves gerenciadas pelo cliente são chaves do KMS em sua Conta da AWS que você cria, detém e gerencia. Você tem controle total sobre essas chaves KMS. É possível:

- Estabelecer e manter suas políticas de chaves, políticas do IAM e concessões
- Habilitá-las e desabilitá-las
- Alternar o material criptográfico
- Adicionar tags
- Criar aliases que se referem a elas
- Agendá-las para exclusão

Para obter mais informações sobre como gerenciar as permissões de uma chave gerenciada pelo cliente, consulte Chaves gerenciadas pelo cliente.

> **Important**
>
> O AWS Glue oferece suporte somente a chaves gerenciadas pelo cliente simétricas. A lista de chaves do KMS exibe apenas chaves simétricas. No entanto, se você selecionar Escolher um ARN de chave do KMS, o console permitirá que você insira um ARN para qualquer tipo de chave. Certifique-se de inserir apenas ARNs para chaves simétricas.

Para criar uma chave simétrica gerenciada pelo cliente, siga as etapas para criar chaves gerenciadas pelo cliente simétricas e o Guia do desenvolvedor do AWS Key Management Service.

Quando você habilita a criptografia do Catálogo de Dados em repouso, os seguintes tipos de recursos são criptografados usando chaves do KMS:

- Bancos de dados
- Tabelas
- Partições
- Versões de tabela
- Estatísticas de colunas
- Funções definidas pelo usuário
- Visualizações do Catálogo de Dados

Contexto de criptografia do AWS Glue

Um contexto de criptografia é um conjunto opcional de pares de chave-valor que pode conter mais informações contextuais sobre os dados. O AWS KMS usa o contexto de criptografia como dados autenticados adicionais para comportar criptografia autenticada. Quando você inclui um contexto de criptografia em uma solicitação para criptografar dados, o AWS KMS vincula de forma criptográfica o contexto de criptografia aos dados criptografados. Para descriptografar dados, é necessário incluir o mesmo contexto de criptografia na solicitação. O AWS Glue usa o mesmo contexto de criptografia em todas as operações criptográficas do AWS KMS, onde a chave é `glue_catalog_id` e o valor é `catalogId`.

```
"encryptionContext": {
    "glue_catalog_id": "111122223333"
}
```

Ao usar uma chave gerenciada pela AWS ou uma chave simétrica gerenciada pelo cliente para criptografar seu Catálogo de Dados, você também pode utilizar o contexto de criptografia em registros de auditoria e logs para identificar como a chave está sendo utilizada. O contexto de criptografia também aparece nos logs gerados pelo AWS CloudTrail ou logs do Amazon CloudWatch.

Habilitar a criptografia

Você pode habilitar a criptografia para seus objetos do AWS Glue Data Catalog nas Configurações do Catálogo de Dados no console do AWS Glue ou usando a AWS CLI.

Console

Para habilitar a criptografia usando o console

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Escolha Catálogo de Dados no painel de navegação.
3. Na página Configurações do Catálogo de Dados, marque a caixa de seleção Criptografia de metadados e escolha uma chave do AWS KMS.

   Quando você habilita a criptografia, se você não especificar uma chave gerenciada pelo cliente, as configurações de criptografia usarão uma chave do KMS gerenciada pela AWS.
4. (Opcional) Quando você usa uma chave gerenciada pelo cliente para criptografar seu Catálogo de Dados, o Catálogo de Dados oferece a opção de registrar um perfil do IAM para criptografar e descriptografar recursos. É necessário conceder ao seu perfil do IAM permissões que o AWS Glue pode assumir em seu nome. Isso inclui permissões do AWS KMS para criptografar e descriptografar dados.

   Quando você cria um novo recurso no Catálogo de Dados, o AWS Glue assume o perfil do IAM fornecido para criptografar os dados. Da mesma forma, quando um consumidor acessa o recurso, o AWS Glue assume o perfil do IAM para descriptografar dados. Se você registrar um perfil do IAM com as permissões necessárias, o responsável pela chamada não precisará mais de permissões para acessar a chave e descriptografar os dados.

   > **Important**
   >
   > É possível delegar operações do KMS a um perfil do IAM somente quando uma chave gerenciada pelo cliente for usada para criptografar os recursos do Catálogo de Dados. No momento, o recurso de delegação de perfil do KMS não oferece suporte ao uso de chaves gerenciadas pela AWS para criptografar recursos do Catálogo de Dados.

> **Warning**
>
> Ao habilitar um perfil do IAM para delegar operações do KMS, você não pode mais acessar os recursos do catálogo de dados que foram criptografados anteriormente com uma chave gerenciada pela AWS.

a. Para habilitar um perfil do IAM que AWS Glue pode assumir para criptografar e descriptografar dados em seu nome, selecione a opção Delegar operações do KMS para um perfil do IAM.

b. Em seguida, escolha um perfil do IAM.

   Para criar um perfil do IAM, consulte Criar um perfil do IAM para o AWS Glue.

   O perfil do IAM que o AWS Glue assume para acessar o Catálogo de Dados deve ter as permissões para criptografar e descriptografar metadados no Catálogo de Dados. Você pode criar um perfil do IAM e anexar as políticas em linha a seguir:

   - Adicione a política a seguir para incluir permissões do AWS KMS para criptografar e descriptografar o Catálogo de Dados.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:<region>:<account-id>:key/<key-id>"
        }
    ]
}
```

   - Em seguida, adicione a política de confiança a seguir ao perfil para que o serviço AWS Glue assuma o perfil do IAM.

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

- Em seguida, adicione a permissão `iam:PassRole` ao perfil do IAM.

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

Ao habilitar a criptografia, se você não tiver especificado um perfil do IAM para o AWS Glue assumir, a entidade principal que acessa o Catálogo de Dados deverá ter permissões para realizar as seguintes operações de API:

- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKey`

AWS CLI

Para habilitar a criptografia usando o SDK ou o AWS CLI

- Use a `PutDataCatalogEncryptionSettings` operação de API. Se nenhuma chave for especificada, o AWS Glue usará a chave de criptografia gerenciada pela AWS para a conta do cliente para criptografar o Catálogo de Dados.

```
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>"
    }

  }'
```

Quando você habilitar a criptografia, todos os objetos que você criar no Catálogo de Dados serão criptografados. Se você desmarcar essa configuração, os objetos que você criar no Catálogo de Dados não serão mais criptografados. Você pode continuar acessando os objetos criptografados existentes no Catálogo de Dados com as permissões do KMS necessárias.

> **Important**
>
> A chave do AWS KMS deve permanecer disponível no armazenamento de chaves do AWS KMS para todos os objetos que são criptografados com ela no Data Catalog. Se você remover a chave, os objetos não poderão mais ser descriptografados. Isso pode ser desejável em alguns cenários para impedir o acesso aos metadados do Data Catalog.

Monitorar suas chaves do KMS para o AWS Glue

Ao usar chaves do KMS com os recursos do Catálogo de Dados, você pode usar o AWS CloudTrail ou os Logs do Amazon CloudWatch para rastrear solicitações enviadas pelo AWS Glue para o AWS

KMS. O AWS CloudTrail monitora e registra as operações do KMS chamadas pelo AWS Glue para acessar dados criptografados por suas chaves do KMS.

Os exemplos a seguir são eventos do AWS CloudTrail para as operações `Decrypt` e `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"
}
```

Criptografar senhas de conexão

Você pode recuperar senhas de conexão no AWS Glue Data Catalog usando o `GetConnection` e `GetConnections` as operações da API. Essas senhas são armazenadas na conexão do Data Catalog e são usadas quando o AWS Glue se conecta a um armazenamento de dados do Java Database Connectivity (JDBC). Se a conexão foi criada ou atualizada, uma opção nas configurações do Data Catalog determina se a senha foi criptografada e, neste caso, qual chave do AWS Key Management Service (AWS KMS) foi especificada.

No console do AWS Glue, é possível ativar essa opção na página Data catalog settings (Configurações do catálogo de dados).

Como criptografas senhas de conexão

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Selecione Configurações no painel de navegação.
3. Na página Data catalog settings (Configurações do catálogo de dados), selecione Encrypt connection passwords (Criptografar senhas de conexão) e escolha uma chave do AWS KMS.

   > **Important**
   >
   > O AWS Glue só oferece suporte a chaves mestras do cliente (CMKs) simétricas. A lista de chaves do AWS KMS exibe apenas chaves simétricas. No entanto, se você selecionar Escolher um ARN de chave AWS KMS, o console permitirá que você insira um ARN para qualquer tipo de chave. Certifique-se de inserir apenas ARNs para chaves simétricas.

   Para ter mais informações, consulte Configurações do Catálogo de Dados .

Criptografar dados gravados pelo AWS Glue

Uma configuração de segurança é um conjunto de propriedades de segurança que pode ser usado pelo AWS Glue. Você pode usar uma configuração de segurança para criptografar dados em repouso. Os cenários a seguir mostram algumas maneiras de usar uma configuração de segurança.

- Anexe uma configuração de segurança a um crawler do AWS Glue para gravar Amazon CloudWatch Logs criptografados. Para obter mais informações sobre como associar configurações de segurança a crawlers, consulte the section called “Etapa 3: Defina as configurações de segurança”.
- Anexe uma configuração de segurança a um trabalho do tipo extrair, transformar e carregar (ETL) para gravar destinos do Amazon Simple Storage Service (Amazon S3) criptografados e CloudWatch Logs criptografados.
- Anexe uma configuração de segurança a um trabalho de ETL para gravar seus marcadores de trabalhos como dados do Amazon S3 criptografados.
- Anexe uma configuração de segurança a um endpoint de desenvolvimento para gravar destinos do Amazon S3 criptografados.

> **Important**
>
> Atualmente, uma configuração de segurança substitui qualquer configuração de criptografia no lado do servidor (SSE-S3) que seja transmitida como um parâmetro de trabalho de ETL. Portanto, se uma configuração de segurança e um parâmetro SSE-S3 forem associados a um trabalho, o parâmetro SSE-S3 será ignorado.

Para obter mais informações sobre configurações de segurança, consulte Trabalhar com configurações de segurança no console do AWS Glue.

Tópicos

- Configurar o AWS Glue para usar configurações de segurança
- Criar uma rota para o AWS KMS para trabalhos e crawlers da VPC
- Trabalhar com configurações de segurança no console do AWS Glue

Configurar o AWS Glue para usar configurações de segurança

Siga estas etapas para configurar seu ambiente AWS Glue para usar configurações de segurança.

1. Crie ou atualize as chaves do AWS Key Management Service (AWS KMS) para conceder permissões ao AWS KMS para as funções do IAM que são transmitidas para crawlers e trabalhos do AWS Glue, a fim de criptografar o CloudWatch Logs. Para obter mais informações,

consulte Criptografar dados de log no CloudWatch Logs usando o AWS KMS no Manual do usuário do Amazon CloudWatch Logs.

No exemplo a seguir, *"role1"*, *"role2"* e *"role3"* são funções do IAM que são transmitidas para crawlers e trabalhos.

```
{
       "Effect": "Allow",
       "Principal": { "Service": "logs.region.amazonaws.com",
       "AWS": [
                "role1",
                "role2",
                "role3"
             ] },
                    "Action": [
                           "kms:Encrypt*",
                           "kms:Decrypt*",
                           "kms:ReEncrypt*",
                           "kms:GenerateDataKey*",
                           "kms:Describe*"
                    ],
                    "Resource": "*"
}
```

A instrução `Service`, mostrada como `"Service": "logs.region.amazonaws.com"`, será necessária se você usar a chave para criptografar o CloudWatch Logs.

2. Certifique-se de que a chave AWS KMS seja `ENABLED` antes de ser usada.

> **Note**
>
> Se você estiver usando o Iceberg como estrutura de data lake, as tabelas do Iceberg têm seus próprios mecanismos para habilitar a criptografia do lado do servidor. Você deve habilitar essas configurações além das configurações de segurança do AWS Glue. Para habilitar a criptografia do lado do servidor nas tabelas do Iceberg, consulte as orientações na documentação do Iceberg.

Criar uma rota para o AWS KMS para trabalhos e crawlers da VPC

Você pode se conectar diretamente ao AWS KMS através de um endpoint privado na sua VPC (virtual private cloud) em vez de fazer a conexão pela Internet. Quando você usa um endpoint da VPC, a comunicação entre seu VPC e o AWS KMS é realizada inteiramente dentro da rede da AWS.

Você pode criar um AWS KMS VPC endpoint dentro de uma VPC. Sem essa etapa, seus trabalhos ou crawlers podem falhar com um `kms timeout` em trabalhos ou com um `internal service exception` em crawlers. Para obter instruções detalhadas, consulte Conectar-se ao AWS KMS por meio de um endpoint da VPC no Guia do desenvolvedor do AWS Key Management Service.

À medida que você seguir estas instruções, no console da VPC, será necessário fazer o seguinte:

- Selecione Enable Private DNS name (Ativar nome de DNS privado).
- Escolha o Grupo de segurança (com regra de autorreferência) que você usa para o seu trabalho ou crawler que acessa o JDBC (Java Database Connectivity). Para obter mais informações sobre conexões do AWS Glue, consulte Conectar a dados.

Quando você adiciona uma configuração de segurança a um crawler ou trabalho que acessa armazenamentos de dados JDBC, o AWS Glue deve ter uma rota para o AWS KMS endpoint. Você pode fornecer essa rota com um gateway de NAT (conversão de endereços de rede) ou com um AWS KMS VPC endpoint. Para criar um gateway NAT, consulte Gateways NAT no Manual do usuário da Amazon VPC.

Trabalhar com configurações de segurança no console do AWS Glue

> **Warning**
>
> As configurações de segurança atuais do AWS Glue não são compatíveis com os trabalhos do Ray.

Uma configuração de segurança no AWS Glue contém as propriedades que são necessárias quando você grava dados criptografados. Você cria configurações de segurança no console do AWS Glue para fornecer as propriedades de criptografia usadas por crawlers, trabalhos e endpoints de desenvolvimento.

Para ver uma lista com todas as configurações de segurança que você criou, abra o console do AWS Glue em https://console.aws.amazon.com/glue/ e escolha Security configurations (Configurações de segurança) no painel de navegação.

A lista de Configurações de segurança exibe as seguintes propriedades sobre cada configuração:

Nome

O nome exclusivo que você forneceu quando criou a configuração. O nome pode conter letras (A a Z), números (0 a 9), hifens (-) ou sublinhados (_), e pode ter até 255 caracteres.

Habilitar a criptografia do Amazon S3

Se ativado, o modo de criptografia do Amazon Simple Storage Service (Amazon S3), como `SSE-KMS` ou `SSE-S3`, é habilitado para armazenamento de metadados no catálogo de dados.

Habilitar a criptografia do Amazon CloudWatch Logs

Se ativado, o modo de criptografia do Amazon S3, como `SSE-KMS`, será habilitado ao gravar logs no Amazon CloudWatch.

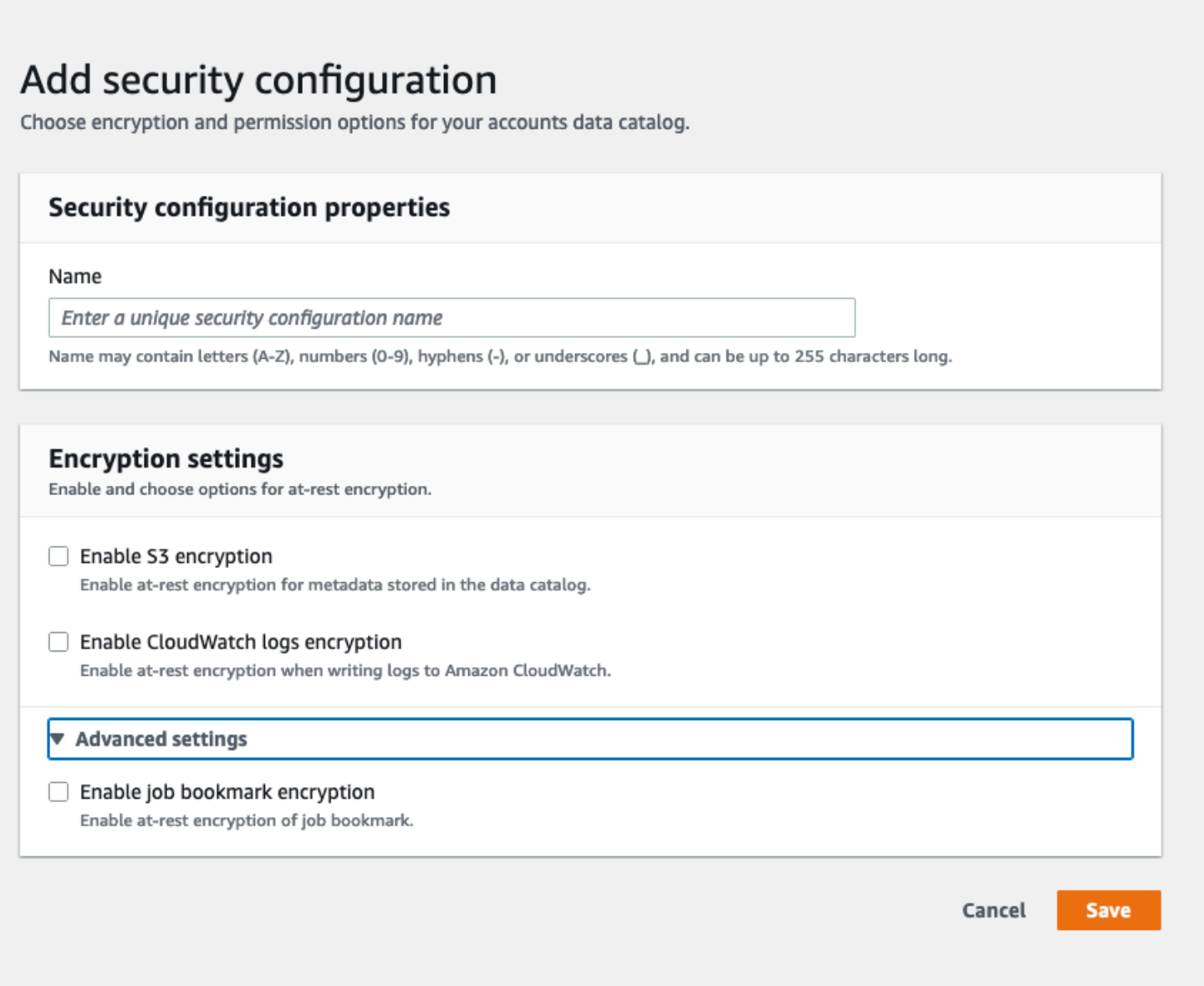
Configurações avançadas: habilitar a criptografia de marcadores de trabalho

Se ativado, o modo de criptografia do Amazon S3, como `CSE-KMS`, será habiltado ao marcar trabalhos.

Você pode adicionar ou excluir configurações na seção Security configurations (Configurações de segurança) do console. Para ver mais detalhes sobre uma configuração, escolha seu nome na lista. Os detalhes incluem as informações que você definiu quando criou a configuração.

Adicionar uma configuração de segurança

Para adicionar uma configuração de segurança usando o console do AWS Glue, na página Security configurations (Configurações de segurança), escolha Add security configuration (Adicionar configuração de segurança).



Propriedades de configuração de segurança

Insira um nome de configuração de segurança exclusivo. O nome pode conter letras (A a Z), números (0 a 9), hifens (-) ou sublinhados (_), e pode ter até 255 caracteres.

Configurações de criptografia

Você pode habilitar a criptografia em repouso para metadados armazenados no catálogo de dados no Amazon S3 e nos logs no Amazon CloudWatch. Para configurar a criptografia de dados e metadados com chaves do AWS Key Management Service (AWS KMS) no console do AWS Glue, adicione uma política ao usuário do console. Essa política deve especificar os recursos

permitidos como nomes de recurso da Amazon (ARNs) de chaves que são usados para criptografar armazenamentos de dados do Amazon S3, como no exemplo a seguir.

```
{
"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**
>
> Quando uma configuração de segurança está anexada a um crawler ou trabalho, a função do IAM que é transmitida deve ter permissões do AWS KMS. Para ter mais informações, consulte Criptografar dados gravados pelo AWS Glue.

Ao definir uma configuração, você pode fornecer valores para as seguintes propriedades:

Habilitar criptografia do S3

Quando você está gravando dados do Amazon S3, pode usar a criptografia no lado do servidor com chaves gerenciadas do Amazon S3 (SSE-S3) ou a criptografia no lado do servidor com chaves gerenciadas do AWS KMS (SSE-KMS). Esse campo é opcional. Para permitir acesso ao Amazon S3, escolha uma chave do AWS KMS ou Enter a key ARN (Inserir o ARN de uma chave), e forneça o ARN da chave. Insira o ARN no formato `arn:aws:kms:region:account-id:key/key-id`. Você também pode fornecer o ARN como um alias da chave, como `arn:aws:kms:region:account-id:alias/alias-name`.

Se você habilitar a interface do Spark para seu trabalho, o arquivo de log da interface do Spark carregado no Amazon S3 será aplicado com a mesma criptografia.

> **Important**
>
> O AWS Glue só oferece suporte a chaves mestras do cliente (CMKs) simétricas. A lista de chaves do AWS KMS exibe apenas chaves simétricas. No entanto, se você selecionar

Escolher um ARN de chave AWS KMS, o console permitirá que você insira um ARN para qualquer tipo de chave. Certifique-se de inserir apenas ARNs para chaves simétricas.

Habilitar criptografia do CloudWatch Logs

A criptografia no lado do servidor (SSE-KMS) é usada para criptografar o CloudWatch Logs. Esse campo é opcional. Para ativá-la, escolha uma chave do AWS KMS ou Enter a key ARN (Inserir ARN da chave), e forneça o ARN da chave. Insira o ARN no formato `arn:aws:kms:`*`region`*`:`*`account-id`*`:key/`*`key-id`*. Você também pode fornecer o ARN como um alias da chave, como `arn:aws:kms:`*`region`*`:`*`account-id`*`:alias/`*`alias-name`*.

Configurações avançadas: criptografia de marcadores de trabalho

A criptografia no lado do cliente (CSE-KMS) é usada para criptografar marcadores de trabalho. Esse campo é opcional. Os dados do marcador são criptografados antes de serem enviados ao Amazon S3 para armazenamento. Para ativá-la, escolha uma chave do AWS KMS ou Enter a key ARN (Inserir ARN da chave), e forneça o ARN da chave. Insira o ARN no formato `arn:aws:kms:`*`region`*`:`*`account-id`*`:key/`*`key-id`*. Você também pode fornecer o ARN como um alias da chave, como `arn:aws:kms:`*`region`*`:`*`account-id`*`:alias/`*`alias-name`*.

Para obter mais informações, consulte os tópicos a seguir no Guia do usuário do Amazon Simple Storage Service:

- Para obter informações sobre o `SSE-S3`, consulte Proteção de dados usando criptografia no lado do servidor com chaves de criptografia gerenciadas do Amazon S3 (SSE-S3).
- Para obter mais informações sobre `SSE-KMS`, consulte Proteger os dados usando criptografia do lado do servidor com AWS KMS keys.
- Para obter informações sobre `CSE-KMS`, consulte Using a KMS key stored in AWS KMS.

Criptografia em trânsito

AWS fornece criptografia Transport Layer Security (TLS) para dados em movimento. Você pode definir as configurações de criptografia para crawlers, trabalhos de ETL e endpoints de desenvolvimento usando configurações de segurança no AWS Glue. É possível ativar a criptografia do AWS Glue Data Catalog por meio das configurações do Data Catalog.

Desde 4 de setembro de 2018, foi adicionado suporte ao AWS KMS (trazer sua própria chave e criptografia no lado do servidor) para o ETL do AWS Glue e o AWS Glue Data Catalog.

Conformidade com os FIPS

Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar a AWS por meio de uma interface de linha de comando ou uma API, use um endpoint do FIPS. Para ter mais informações sobre endpoints do FIPS, consulte Federal Information Processing Standard (FIPS) 140-2.

Gerenciamento de chaves

É possível usar o AWS Identity and Access Management (IAM) com o AWS Glue para definir usuários, recursos da AWS, grupos, funções e políticas granulares com relação a acesso, negação e muito mais.

Você pode definir o acesso aos metadados usando as políticas baseadas em recursos e as políticas baseadas em identidade dependendo das necessidades da sua organização. As políticas baseadas em recursos listam as entidades principais com acesso permitido ou negado aos seus recursos, permitindo que você configure políticas como acesso entre contas. As políticas baseadas em identidade são anexadas de modo exclusivo a usuários, grupos e funções no IAM.

Para obter um exemplo passo a passo, consulte Restringir o acesso ao seu AWS Glue Data Catalog com permissões de IAM de nível de recursos e políticas baseadas em recursos no AWS Big Data Blog.

A parte de acesso granular da política é definida na cláusula `Resource`. Essa parte define o objeto do AWS Glue Data Catalog no qual a ação pode ser executada e quais objetos resultantes são retornados por essa operação.

Um endpoint de desenvolvimento é um ambiente que você pode usar para desenvolver e testar seus scripts do AWS Glue. É possível adicionar, excluir ou mudar a chave SSH de um endpoint de desenvolvimento.

Desde 4 de setembro de 2018, foi adicionado suporte ao AWS KMS (trazer sua própria chave e criptografia no lado do servidor) para o ETL do AWS Glue e o AWS Glue Data Catalog.

Dependência do AWS Glue de outros produtos da AWS

Para que um usuário trabalhe com o console do AWS Glue, ele deve ter um conjunto mínimo de permissões que liberem o trabalho com os recursos do AWS Glue na conta da AWS. Além dessas permissões do AWS Glue, o console exige permissões dos seguintes serviços:

- Permissões do Amazon CloudWatch Logs para exibir logs.
- Permissões do AWS Identity and Access Management (IAM) para listar e transmitir funções.
- Permissões do AWS CloudFormation para trabalhar com pilhas.
- Permissões do Amazon Elastic Compute Cloud (Amazon EC2) para listar nuvens privadas virtuais (VPCs), sub-redes, grupos de segurança, instâncias e outros objetos (para configurar itens do Amazon EC2, como VPCs, ao executar trabalhos, crawlers e criar endpoints de desenvolvimento).
- Permissões do Amazon Simple Storage Service (Amazon S3) para listar buckets e objetos e para recuperar e salvar scripts.
- Permissões do Amazon Redshift necessárias para trabalhar com clusters.
- Permissões do Amazon Relational Database Service (Amazon RDS) para listar instâncias.

Endpoints de desenvolvimento

Um endpoint de desenvolvimento é um ambiente que você pode usar para desenvolver e testar seus scripts do AWS Glue. Você pode usar o AWS Glue para criar, editar e excluir endpoints de desenvolvimento. Você pode listar todos os endpoints de desenvolvimento criados. É possível adicionar, excluir ou mudar a chave SSH de um endpoint de desenvolvimento. Também é possível criar blocos de anotações que usam o endpoint de desenvolvimento.

Você fornece valores de configuração para a provisão dos ambientes de desenvolvimento. Esses valores informam ao AWS Glue como configurar a rede para que você possa acessar o endpoint de desenvolvimento de forma segura e para que seu endpoint possa acessar seus armazenamentos de dados. Depois, você poderá criar um bloco de anotações que se conecta ao endpoint de desenvolvimento. Você usará o bloco de anotações para criar e testar seu script de ETL.

Escolha uma função do AWS Identity and Access Management (IAM) com permissões semelhantes à função do IAM que você usa para executar os trabalhos de ETL do AWS Glue. Use uma nuvem privada virtual (VPC), uma sub-rede e um grupo de segurança para criar um endpoint de desenvolvimento que possa se conectar a seus recursos de dados com segurança. Você gera um par de chaves SSH para se conectar ao ambiente de desenvolvimento usando SSH.

É possível criar endpoints de desenvolvimento para dados do Amazon S3 e em uma VPC, que você pode usar para acessar conjuntos de dados usando o JDBC.

Você pode instalar um caderno Jupyter na sua máquina local e usá-lo para depurar e testar scripts de ETL em um endpoint de desenvolvimento. Ou você pode usar um notebook do Sagemaker para criar scripts de ETL no JupyterLab na AWS. Consulte Usar um caderno do SageMaker com seu endpoint de desenvolvimento .

O AWS Glue marca instâncias do Amazon EC2 com um nome que é prefixado com `aws-glue-dev-endpoint`.

Você pode configurar um servidor de caderno em um endpoint de desenvolvimento para executar instruções do PySpark com as extensões do AWS Glue.

Gerenciamento de identidade e acesso para AWS Glue

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar os recursos do AWS Glue. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

> **Note**
>
> Você pode conceder acesso aos seus dados no AWS Glue Data Catalog usando AWS Glue métodos ou AWS Lake Formation concessões. Você pode usar políticas AWS Identity and Access Management (IAM) para definir um controle de acesso refinado com métodos. AWS Glue O Lake Formation usa um modelo de permissões `GRANT/REVOKE` mais simples, semelhante aos comandos `GRANT/REVOKE` de um sistema de banco de dados relacional. Esta seção inclui informações sobre como usar os métodos do AWS Glue. Para obter mais informações sobre o uso das concessões do Lake Formation, consulte Conceder permissões do Lake Formation no Guia do desenvolvedor do AWS Lake Formation .

Tópicos

- Público
- Autenticando com identidades

- Gerenciando acesso usando políticas
- Como o AWS Glue funciona com o IAM
- Configurar permissões do KMS para o AWS Glue
- Exemplos de política de controle de acesso do AWS Glue
- AWS políticas gerenciadas para AWS Glue
- Especificação de ARNs de recursos do AWS Glue
- Concessão de acesso entre contas
- Solução de problemas de identidade e acesso ao AWS Glue

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no AWS Glue.

Usuário do serviço — Se você usa o serviço AWS Glue para fazer seu trabalho, seu administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais recursos do AWS Glue para fazer seu trabalho, talvez precise de permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao seu administrador. Se você não puder acessar um recurso no AWS Glue, consulte Solução de problemas de identidade e acesso ao AWS Glue.

Administrador de serviços — Se você é responsável pelos recursos do AWS Glue em sua empresa, provavelmente tem acesso total ao AWS Glue. É seu trabalho determinar quais recursos e recursos do AWS Glue seus usuários do serviço devem acessar. Assim, você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como sua empresa pode usar o IAM com o AWS Glue, consulteComo o AWS Glue funciona com o IAM.

Administrador do IAM — Se você for administrador do IAM, talvez queira saber detalhes sobre como criar políticas para gerenciar o acesso ao AWS Glue. Para ver exemplos de políticas baseadas em identidade do AWS Glue que você pode usar no IAM, consulte. Exemplos das políticas baseadas em identidade do AWS Glue

Autenticando com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte Assinatura de solicitações de AWS API no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação Multifator no Guia do Usuário do AWS IAM Identity Center . Usar a autenticação multifator (MFA) na AWS no Guia do Usuário do IAM.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do Usuário do IAM.

Identidade federada

Como prática recomendada, exija que usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, de um provedor de identidade da web AWS Directory Service, do diretório do Identity Center ou de qualquer usuário que acesse usando credenciais fornecidas Serviços da AWS por meio de uma fonte de identidade. Quando as identidades federadas são acessadas Contas da AWS, elas assumem funções, e as funções fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todos os seus Contas da AWS aplicativos. Para obter mais informações sobre o Centro de Identidade do IAM, consulte O que é o Centro de Identidade do IAM? no Manual do Usuário do AWS IAM Identity Center .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do Usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de um perfil) no Guia do usuário do IAM.

Perfis do IAM

Uma função do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Você pode assumir temporariamente uma função do IAM no AWS Management Console trocando de funções. Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para o uso de perfis, consulte Utilizar perfis do IAM no Guia do usuário do IAM.

Funções do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidades de terceiros no Guia do Usuário do IAM. Se você usar o Centro de identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no Manual do Usuário do AWS IAM Identity Center .
- Permissões temporárias para usuários do IAM –- um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas –- é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service

(Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.

- Função de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.
- Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para funções vinculadas ao serviço.

- Aplicativos em execução no Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma instância do EC2 e fazendo AWS CLI solicitações de API. AWS É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma AWS função a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Utilizar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar perfis do IAM, consulte Quando criar um perfil do IAM (em vez de um usuário) no Guia do usuário do IAM.

Gerenciando acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do Usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissões para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem presumir os perfis.

As políticas do IAM definem permissões para uma ação independente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do Usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes as políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o Amazon VPC são exemplos de serviços que oferecem suporte a ACLs. AWS WAF Para saber mais sobre ACLs, consulte Visão geral da lista de controle de acesso (ACL) no Guia do Desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um atributo avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do Usuário do IAM.
- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em. AWS Organizations AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizações e SCPs, consulte Como os SCPs Funcionam no Manual do Usuário do AWS Organizations .
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do

usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do Usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como o AWS Glue funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao AWS Glue, saiba quais recursos do IAM estão disponíveis para uso com o AWS Glue.

Recursos do IAM que você pode usar com o AWS Glue

Atributo do IAM	AWS Suporte Glue
Políticas baseadas em identidade	Sim
Políticas baseadas em atributos	Parcial
Ações das políticas	Sim
Atributos de políticas	Sim
Chaves de condição de política (específicas do serviço)	Sim
ACLs	Não
ABAC (tags em políticas)	Parcial
Credenciais temporárias	Sim
Permissões de entidade principal	Não

Atributo do IAM	AWS Suporte Glue
Perfis de serviço	Sim
Perfis vinculados ao serviço	Não

Para ter uma visão de alto nível de como o AWS Glue e outros AWS serviços funcionam com a maioria dos recursos do IAM, consulte AWS os serviços que funcionam com o IAM no Guia do usuário do IAM.

Políticas baseadas em identidade para Glue AWS

Suporta políticas baseadas em identidade	Sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações ou recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou perfil ao qual ela está anexada. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elementos da política JSON do IAM no Guia do Usuário do IAM.

O AWS Glue oferece suporte a políticas baseadas em identidades (políticas do IAM) para todas as operações do AWS Glue. Anexando uma política, você pode conceder permissões para criar, acessar ou modificar um recurso do AWS Glue, como uma tabela no AWS Glue Data Catalog.

Exemplos de políticas baseadas em identidade para Glue AWS

Para ver exemplos de políticas baseadas em identidade do AWS Glue, consulte. Exemplos das políticas baseadas em identidade do AWS Glue

Políticas baseadas em recursos no Glue AWS

Oferece compatibilidade com políticas baseadas em recursos	Parcial

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, você pode especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em atributo. Adicionar uma entidade principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso são diferentes Contas da AWS, um administrador do IAM na conta confiável também deve conceder permissão à entidade principal (usuário ou função) para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a uma entidade principal na mesma conta, nenhuma política baseada em identidade adicional será necessária. Para obter mais informações, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

> **Note**
>
> Uma política de recursos do AWS Glue só pode ser usada para gerenciar permissões para recursos do Data Catalog. Não é possível anexá-la a todos os outros recursos do AWS Glue, como trabalhos, gatilhos, endpoints de desenvolvimento, crawlers ou classificadores. Apenas uma política de recursos é permitida por catálogo, e seu tamanho é limitado a 10 KB.

No AWS Glue, uma política de recursos é anexada a um catálogo, que é um contêiner virtual para todos os tipos de recursos do Catálogo de Dados mencionados anteriormente. Cada AWS conta possui um único catálogo em uma AWS região cujo ID do catálogo é igual ao ID da AWS conta. Você não pode excluir nem modificar um catálogo.

Uma política de recurso é avaliada para todas as chamadas da API ao catálogo em que a entidade principal do chamador é incluída no bloco `"Principal"` do documento da política.

Para ver exemplos de políticas baseadas em recursos do AWS Glue, consulte. Exemplos de política baseada em recursos para o AWS Glue

Ações políticas para AWS Glue

Oferece compatibilidade com ações de políticas	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Action` de uma política JSON descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da operação de AWS API associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de ações do AWS Glue, consulte Ações definidas pelo AWS Glue na Referência de Autorização de Serviço.

As ações de política no AWS Glue usam o seguinte prefixo antes da ação:

```
glue
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "glue:action1",
      "glue:action2"
         ]
```

Você também pode especificar várias ações usando caracteres-curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Get`, inclua a seguinte ação:

```
"Action": "glue:Get*"
```

Para ver exemplos de políticas, consulte Exemplos de política de controle de acesso do AWS Glue.

Recursos de políticas para AWS Glue

Oferece compatibilidade com recursos de políticas	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para obter mais informações sobre como controlar o acesso aos recursos do AWS Glue usando ARNs, consulteEspecificação de ARNs de recursos do AWS Glue.

Para ver uma lista dos tipos de recursos do AWS Glue e seus ARNs, consulte Recursos definidos pelo AWS Glue na Referência de Autorização de Serviço. Para saber quais ações você pode usar para especificar o ARN de cada recurso, consulte Ações definidas pelo AWS Glue.

Chaves de condição de política para AWS Glue

Suporta chaves de condição de política específicas de serviço	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Condition` (ou bloco `Condition`) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos `Condition` em uma instrução ou várias chaves em um único `Condition` elemento, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um atributo somente se ele estiver marcado com seu nome de usuário do IAM. Para obter mais informações, consulte Elementos da política do IAM: variáveis e tags no Guia do usuário do IAM.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

Para ver uma lista das chaves de condição do AWS Glue, consulte Chaves de condição do AWS Glue na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte Ações definidas pelo AWS Glue.

Para ver exemplos de políticas, consulte Controlar as configurações usando chaves de condição ou chaves de contexto.

ACLs em Glue AWS

Oferece compatibilidade com ACLs	Não

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes as políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

ABAC com AWS Glue

Oferece compatibilidade com ABAC (tags em políticas)	Parcial

O controle de acesso baseado em recurso (ABAC) é uma estratégia de autorização que define permissões com base em recursos. Em AWS, esses atributos são chamados de tags. Você pode anexar tags a entidades do IAM (usuários ou funções) e a vários AWS recursos. A marcação de entidades e atributos é a primeira etapa do ABAC. Em seguida, você cria políticas de ABAC para permitir operações quando a tag da entidade principal corresponder à tag do recurso que ela estiver tentando acessar.

O ABAC é útil em ambientes que estão crescendo rapidamente e ajuda em situações onde o gerenciamento de políticas se torna um problema.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`.

Se um serviço oferecer suporte às três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço oferecer suporte às três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para obter mais informações sobre o ABAC, consulte O que é ABAC? no Guia do Usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Utilizar controle de acesso baseado em atributos (ABAC) no Guia do usuário do IAM.

> **Important**
>
> As chaves de contexto de condição se aplicam somente às ações de API do AWS Glue em crawlers, trabalhos, triggers e endpoints de desenvolvimento. Para obter mais informações sobre quais operações de API são afetadas, consulte Chaves de condição para AWS Glue. Atualmente, as operações de API do AWS Glue Data Catalog não são compatíveis com as chaves globais de contexto de condição `aws:referer` e `aws:UserAgent`.

Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um recurso baseado em tags desse recurso, consulte Conceder acesso usando tags.

Usando credenciais temporárias com AWS Glue

Oferece compatibilidade com credenciais temporárias	Sim

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS funcionam com credenciais temporárias, consulte Serviços da AWS “Trabalhe com o IAM” no Guia do usuário do IAM.

Você está usando credenciais temporárias se fizer login AWS Management Console usando qualquer método, exceto um nome de usuário e senha. Por exemplo, quando você acessa AWS usando o link de login único (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna perfis. Para obter mais informações sobre como alternar funções, consulte Alternar para um perfil (console) no Guia do usuário do IAM.

Você pode criar manualmente credenciais temporárias usando a AWS API AWS CLI ou. Em seguida, você pode usar essas credenciais temporárias para acessar AWS. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para mais informações, consulte Credenciais de segurança temporárias no IAM.

Permissões principais entre serviços para AWS Glue

Suporte para o recurso Encaminhamento de sessões de acesso (FAS)	Não

Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.

Perfis de serviço do AWS Glue

Oferece compatibilidade com funções de serviço	Sim

O perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

> **Warning**
>
> Alterar as permissões de uma função de serviço pode interromper a funcionalidade do AWS Glue. Edite as funções de serviço somente quando o AWS Glue fornecer orientação para fazer isso.

Para obter instruções detalhadas sobre como criar uma função de serviço para o AWS Etapa 1: criar uma política do IAM para o serviço AWS Glue Glue, consulte Etapa 2: criar um perfil do IAM para o AWS Glue e.

Funções vinculadas a serviços para Glue AWS

Oferece suporte a perfis vinculados ao serviço	Não

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar perfis vinculados a serviços, consulte Serviços da AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Função vinculada ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Configurar permissões do KMS para o AWS Glue

Você usa o AWS Identity and Access Management (IAM) para definir as políticas e os perfis que o AWS Glue usa para acessar os recursos. As etapas a seguir orientam você por várias opções para configurar as permissões para o AWS Glue. Dependendo das necessidades da sua empresa, pode ser necessário adicionar ou reduzir o acesso aos seus recursos.

> **Note**
>
> Para começar a usar as permissões básicas do IAM para o AWS Glue em vez disso, consulte Configurar permissões do IAM para o AWS Glue.

1. Criar uma política do IAM para o serviço AWS Glue: crie uma política de serviço que conceda acesso aos recursos do AWS Glue.
2. Criar um perfil do IAM para o AWS Glue: crie um perfil do IAM e anexe a política do serviço AWS Glue e uma política para os recursos do Amazon Simple Storage Service (Amazon S3) que são usados pelo AWS Glue.
3. Anexar uma política a usuários ou grupos que acessam o AWS Glue: anexe políticas aos usuários ou grupos que fazem login no console do AWS Glue.
4. Criar uma política do IAM para cadernos: crie uma política de servidor de cadernos para usar na criação de servidores de cadernos em endpoints de desenvolvimento.
5. Criar um perfil do IAM para cadernos: crie um perfil do IAM e anexe a política do servidor do cadernos.
6. Criar uma política do IAM para cadernos do Amazon SageMaker: crie uma política do IAM a ser usada ao criar cadernos do Amazon SageMaker em endpoints de desenvolvimento.
7. Criar um perfil do IAM para cadernos do Amazon SageMaker: crie um perfil do IAM e anexe a política para conceder permissões ao criar cadernos do Amazon SageMaker em endpoints de desenvolvimento.

Etapa 1: criar uma política do IAM para o serviço AWS Glue

Para qualquer operação que acesse dados em outro recurso da AWS, como o acesso aos seus objetos no Amazon S3,o AWS Glue precisa de permissão para acessar o recurso em seu nome. Você concede essas permissões usando o AWS Identity and Access Management (IAM).

> **Note**
>
> Você pode ignorar esta etapa se usar a política gerenciada pela AWS **`AWSGlueServiceRole`**.

Nesta etapa, você cria uma política semelhante a `AWSGlueServiceRole`. Você pode encontrar a versão mais atual da `AWSGlueServiceRole` no console do IAM.

Para criar uma política do IAM para AWS Glue

Essa política concede permissão a algumas ações do Amazon S3 para o gerenciamento dos recursos na sua conta que são exigidos pelo AWS Glue quando ele assume uma função usando essa política. Alguns dos recursos especificados nessa política referem-se a nomes padrão que são usados pelo AWS Glue para buckets do Amazon S3, scripts de ETL do Amazon S3, CloudWatch Logs e recursos do Amazon EC2. Por questões de simplicidade, o AWS Glue grava alguns objetos do Amazon S3 em buckets da sua conta com o prefixo `aws-glue-*` por padrão.

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Políticas.
3. Escolha Create Policy (Criar política).
4. Na tela Create Policy, navegue até uma guia para editar o JSON. Crie um documento de política com as seguintes instruções JSON e escolha Review policy.

   > **Note**
   >
   > Adicione quaisquer permissões necessárias para os recursos do Amazon S3. Convém definir o escopo da seção de recursos da sua política de acesso somente para os recursos necessários.

```
{
    "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/*"
            ]
        }
    ]
}
```

A tabela a seguir descreve as permissões concedidas por esta política.

Ação	Recurso	Descrição
`"glue:*"`	`"*"`	Concede permissão para executar todas as operações de API do AWS Glue.
`"s3:GetBucketLocation", "s3:ListBucket", "s3:ListAllMyBuckets", "s3:GetBucketAcl",`	`"*"`	Permite listar os buckets do Amazon S3 a partir de crawlers, trabalhos, endpoints de desenvolvimento e servidores de cadernos.
`"ec2:DescribeVpcEndpoints", "ec2:DescribeRouteTables", "ec2:CreateNetworkInterface", "ec2:DeleteNetworkInterface", "ec2:DescribeNetworkInterfaces", "ec2:DescribeSecurityGroups", "ec2:DescribeSubnets", "ec2:DescribeVpcAttribute",`	`"*"`	Permite a configuração de itens de rede do Amazon EC2, como nuvens privadas virtuais (VPCs), ao executar trabalhos, crawlers e endpoints de desenvolvimento.
`"iam:ListRolePolicies", "iam:GetRole", "iam:GetRolePolicy"`	`"*"`	Permite listar as funções do IAM a partir de crawlers, trabalhos, endpoints de desenvolvimento e servidores de cadernos.

Ação	Recurso	Descrição
`"cloudwatch:PutMetricData"`	`"*"`	Permite gravar métricas do CloudWatch para trabalhos.
`"s3:CreateBucket", "s3:PutBucketPublicAccessBlock"`	`"arn:aws:s3:::aws-glue-*"`	Permite criar buckets do Amazon S3 na sua conta a partir de trabalhos e servidores de cadernos. Convenção de nomenclatura: usa pastas do Amazon S3 chamadas aws-glue-. Permite que o AWS Glue crie buckets que bloqueiam o acesso público.
`"s3:GetObject", "s3:PutObject", "s3:DeleteObject"`	`"arn:aws:s3:::aws-glue-*/*", "arn:aws:s3:::*/*aws-glue-*/*"`	Possibilita obter, inserir e excluir objetos do Amazon S3 contidos na sua conta ao armazenar objetos, como scripts de ETL e locais de servidores de cadernos. Convenção de nomenclatura: concede permissão a buckets ou pastas do Amazon S3 cujos nomes contêm o prefixo aws-glue-.

Ação	Recurso	Descrição
`"s3:GetObject"`	`"arn:aws:s3:::crawler-public*", "arn:aws:s3:::aws-glue-*"`	Permite obter objetos do Amazon S3 usados por exemplos e tutoriais dos crawlers e trabalhos. Convenção de nomenclatura: os nomes de buckets do Amazon S3 começam com crawler-public e aws-glue-.
`"logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents"`	`"arn:aws:logs:*:*:log-group:/aws-glue/*"`	Permite gravar logs no CloudWatch Logs. Convenção de nomenclatura: o AWS Glue grava logs em grupos cujos nomes começam com aws-glue.
`"ec2:CreateTags", "ec2:DeleteTags"`	`"arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:instance/*"`	Permite marcar de recursos do Amazon EC2 criados para endpoints de desenvolvimento. Convenção de nomenclatura: o AWS Glue marca interfaces de rede, grupos de segurança e instâncias do Amazon EC2 com aws-glue-service-resource.

5. Na tela Review Policy (Revisar política), insira o Policy Name (Nome da política), por exemplo, GlueServiceRolePolicy. Digite uma descrição opcional e, quando estiver satisfeito com a política, escolha Create policy (Criar política).

Etapa 2: criar um perfil do IAM para o AWS Glue

Você precisa conceder à sua função do IAM as permissões que o AWS Glue pode assumir ao chamar outros serviços em seu nome. Isso inclui acesso do Amazon S3 a quaisquer fontes, destinos, scripts e diretórios temporários utilizados com o AWS Glue. Crawlers, trabalhos e endpoints de desenvolvimento precisam dessa permissão.

Você concede essas permissões usando o AWS Identity and Access Management (IAM). Adicione uma política à função do IAM que você transmitir ao AWS Glue.

Para criar um perfil do IAM para o AWS Glue

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.
3. Selecione Criar função.
4. Em tipo de perfil, escolha Serviço da AWS, localize e escolha Glue e depois escolha Próximo: permissões.
5. Na página Attach permissions policy (Anexar política de permissões), escolha as políticas que contenham as permissões necessárias. Por exemplo, a política gerenciadas pela AWS `AWSGlueServiceRole` para permissões gerais do AWS Glue e a política gerenciada pela AWS AmazonS3FullAccess para acesso aos recursos do Amazon S3. Então, escolha Próximo: Análise.

> **Note**
>
> Verifique se uma das políticas nessa função concede permissões aos seus destinos e fontes do Amazon S3. Convém fornecer sua própria política de acesso a recursos específicos do Amazon S3. As fontes de dados exigem permissões `s3:ListBucket` e `s3:GetObject`. Os destinos de dados exigem permissões `s3:ListBucket`, `s3:PutObject` e `s3:DeleteObject`. Para obter mais informações sobre como criar uma política do Amazon S3 para os seus recursos, consulte Especificar recursos em uma política. Para obter um exemplo de política do Amazon S3, consulte Gravar políticas do IAM: como conceder acesso a um bucket do Amazon S3.
> Se você pretende acessar as fontes e os destinos do Amazon S3 que foram criptografados com SSE-KMS, anexe uma política que permita que os crawlers, os trabalhos e os endpoints de desenvolvimento do AWS Glue descriptografem os dados.

Para obter mais informações, consulte Proteção de dados usando criptografia do lado do servidor com chaves gerenciadas pelo AWS KMS (SSE-KMS).
Veja um exemplo a seguir.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
         ]
      }
   ]
}
```

6. Para Role name (Nome da função), digite um nome para sua função, por exemplo, `AWSGlueServiceRoleDefault`. Crie a função com um nome que tenha a string `AWSGlueServiceRole` como prefixo para permitir que a função seja transmitida dos usuários do console para o serviço. As políticas fornecidas pelo AWS Glue esperam que as funções de serviço do IAM comecem com `AWSGlueServiceRole`. Caso contrário, você precisará adicionar uma política para conceder aos seus usuários a permissão `iam:PassRole` para que as funções do IAM correspondam às suas conversões de nomenclatura. Selecione Criar perfil.

> **Note**
>
> Quando você cria um caderno com uma função, essa função é transmitida para sessões interativas, de modo que a mesma função possa ser usada em ambos os lugares. Por isso, a permissão `iam:PassRole` precisa fazer parte da política da função.
> Crie uma nova política para seu perfil usando o exemplo a seguir. Substitua o número da conta e pelo seu próprio e pelo nome do perfil.
>
> ```
> {
>     "Version": "2012-10-17",
>     "Statement": [
>         {
>             "Effect": "Allow",
> ```

```
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::090000000210:role/<role_name>"
        }
    ]
}
```

Etapa 3: anexar uma política aos usuários ou grupos que acessam o AWS Glue

O administrador deve atribuir permissões a todos os usuários, grupos ou perfis que usem o console do AWS Glue ou a AWS Command Line Interface (AWS CLI). Você fornece essas permissões usando o AWS Identity and Access Management (IAM), por meio de políticas. Esta etapa descreve a atribuição de permissões a usuários ou grupos.

Quando você concluir esta etapa, o usuário ou o grupo terá as seguintes políticas anexadas:

- A política gerenciada pela AWS `AWSGlueConsoleFullAccess` ou a política personalizada GlueConsoleAccessPolicy
- **`AWSGlueConsoleSageMakerNotebookFullAccess`**
- **`CloudWatchLogsReadOnlyAccess`**
- **`AWSCloudFormationReadOnlyAccess`**
- **`AmazonAthenaFullAccess`**

Para anexar uma política em linha e incorporá-la em um usuário ou grupo

Você pode anexar uma política gerenciada pela AWS ou uma política em linha a um usuário ou grupo para acessar o console do AWS Glue. Alguns dos recursos especificados nessa política referem-se a nomes padrão que são usados pelo AWS Glue para buckets do Amazon S3, scripts de ETL do Amazon S3, CloudWatch Logs, AWS CloudFormation e recursos do Amazon EC2. Por questões de simplicidade, o AWS Glue grava alguns objetos do Amazon S3 em buckets da sua conta com o prefixo `aws-glue-*` por padrão.

> **Note**
>
> Você pode ignorar esta etapa se usar a política gerenciada pela AWS **`AWSGlueConsoleFullAccess`**.

> **Important**
>
> O AWS Glue precisa de permissão para assumir uma função usada para executar um trabalho em seu nome. Para fazer isso, você adiciona as permissões de **`iam:PassRole`** aos usuários ou aos grupos do AWS Glue. Essa política concede permissão às funções que começam com `AWSGlueServiceRole` para funções de serviço do AWS Glue e com `AWSGlueServiceNotebookRole` para funções necessárias ao criar um servidor de notebook. Você também pode criar sua própria política para permissões `iam:PassRole` seguindo sua convenção de nomenclatura.
> De acordo com as práticas recomendadas de segurança, recomenda-se restringir o acesso enrijecendo políticas para restringir ainda mais o acesso ao bucket do Amazon S3 e grupos de logs do Amazon CloudWatch. Para obter um exemplo de política do Amazon S3, consulte Gravar políticas do IAM: como conceder acesso a um bucket do Amazon S3.

Nesta etapa, você cria uma política semelhante a `AWSGlueConsoleFullAccess`. Você pode encontrar a versão mais atual da `AWSGlueConsoleFullAccess` no console do IAM.

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Grupos ou Usuários.
3. Na lista, escolha o nome do usuário ou do grupo ao qual deseja incorporar uma política.
4. Selecione a guia Permissions (Permissões) e expanda a seção Permissions policies (Políticas de permissões).
5. Escolha o link Add Inline policy.
6. Na tela Create Policy, navegue até uma guia para editar o JSON. Crie um documento de política com as seguintes instruções JSON e escolha Review policy.

```
{
    "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"
                    ]
                }
            }
        }
    ]
}
```

A tabela a seguir descreve as permissões concedidas por esta política.

Ação	Recurso	Descrição
`"glue:*"`	`"*"`	Concede permissão para executar todas as operações de API do AWS Glue. Se você criou sua política anteriormente sem a ação `"glue:*"`, é necessário adicionar as seguintes permissões individuais à sua política: • "glue:ListCrawlers" • "glue:BatchGetCrawlers" • "glue:ListTriggers" • "glue:BatchGetTriggers" • "glue:ListDevEndpoints" • "glue:BatchGetDevEndpoints" • "glue:ListJobs" • "glue:BatchGetJobs"
`"redshift:DescribeClusters", "redshift:DescribeClusterSubnetGroups"`	`"*"`	Permite criar conexões com o Amazon RedShift.

Ação	Recurso	Descrição
"iam:ListRoles", "iam:ListRolePolicies", "iam:GetRole", "iam:GetRolePolicy", "iam:ListAttachedRolePolicies"	"*"	Permite listar funções do IAM ao trabalhar com crawlers, trabalhos, endpoints de desenvolvimento e servidores de cadernos.
"ec2:DescribeSecurityGroups", "ec2:DescribeSubnets", "ec2:DescribeVpcs", "ec2:DescribeVpcEndpoints", "ec2:DescribeRouteTables", "ec2:DescribeVpcAttribute", "ec2:DescribeKeyPairs", "ec2:DescribeInstances"	"*"	Permite a configuração dos itens de rede do Amazon EC2 (como VPCs) ao executar trabalhos, crawlers e endpoints de desenvolvimento.
"rds:DescribeDBInstances"	"*"	Permite criar conexões com o Amazon RDS.
"s3:ListAllMyBuckets", "s3:ListBucket", "s3:GetBucketAcl", "s3:GetBucketLocation"	"*"	Permite listar buckets do Amazon S3 ao trabalhar com crawlers, trabalhos, endpoints de desenvolvimento e servidores de cadernos.
"dynamodb:ListTables"	"*"	Permite listar tabelas do DynamoDB.
"kms:ListAliases", "kms:DescribeKey"	"*"	Permite trabalhar com chaves KMS.

Ação	Recurso	Descrição
`"cloudwatch:GetMetricData", "cloudwatch:ListDashboards"`	`"*"`	Permite trabalhar com métricas do CloudWatch.
`"s3:GetObject", "s3:PutObject"`	`"arn:aws:s3::: aws-glue-*/*", "arn:aws:s3::: */*aws-glue-*/*", "arn:aws:s3::: aws-glue-*"`	Possibilita obter e inserir objetos do Amazon S3 contidos na sua conta ao armazenar objetos, como scripts de ETL e locais de servidores de cadernos. Convenção de nomenclatura: concede permissão a buckets ou pastas do Amazon S3 cujos nomes contêm o prefixo aws-glue-.
`"tag:GetResources"`	`"*"`	Permite a recuperação de tags da AWS.

Ação	Recurso	Descrição
`"s3:CreateBucket", "s3:PutBucketPublicAccessBlock"`	`"arn:aws:s3::: aws-glue-*"`	Possibilita a criação de um bucket do Amazon S3 na sua conta ao armazenar objetos, como scripts de ETL e locais de servidores de cadernos. Convenção de nomenclatura: concede permissão a buckets do Amazon S3 cujos nomes contêm o prefixo aws-glue-. Permite que o AWS Glue crie buckets que bloqueiam o acesso público.
`"logs:GetLogEvents"`	`"arn:aws:logs:*:*: / aws-glue/*"`	Permite recuperar CloudWatch Logs. Convenção de nomenclatura: o AWS Glue grava logs em grupos cujos nomes começam com aws-glue.

Ação	Recurso	Descrição
`"cloudformation:CreateStack", "cloudformation:DeleteStack"`	`"arn:aws:cloudformation:*:*:stack/aws-glue*/*"`	Permite o gerenciamento de pilhas do AWS CloudFormation ao trabalhar com servidores de notebook. Convenção de nomenclatura: o AWS Glue cria pilhas cujos nomes começam com 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 a execução de endpoints de desenvolvimento e de servidores de notebook.
`"iam:PassRole"`	`"arn:aws:iam::*:role/ AWSGlueServiceRole*"`	Permite que o AWS Glue assuma permissões `PassRole` de regras que começam com `AWSGlueServiceRole` .

Ação	Recurso	Descrição
`"iam:PassRole"`	`"arn:aws:iam::*:role/ AWSGlueServiceNotebookRole*"`	Permite que o Amazon EC2 assuma permissões `PassRole` de regras que começam com `AWSGlueServiceNotebookRole` .
`"iam:PassRole"`	`"arn:aws:iam::*:role/service-role/AWSGlueServiceRole*"`	Permite que o AWS Glue assuma permissões `PassRole` de regras que começam com `service-role/AWSGlueServiceRole` .

7. Na página Review policy (Revisar política), insira um nome para a política, por exemplo, GlueConsoleAccessPolicy. Quando estiver satisfeito com a política, escolha Create policy (Criar política). Certifique-se de que nenhum erro seja exibido na caixa vermelha na parte superior da tela. Corrija os que foram relatados.

> **Note**
>
> Se a opção Use autoformatting estiver selecionada, a política será reformatada sempre que você abrir uma política ou escolher Validate Policy.

Para anexar a política gerenciada AWSGlueConsoleFullAccess

Você pode anexar a política `AWSGlueConsoleFullAccess` para fornecer permissões exigidas pelo usuário do console do AWS Glue.

> **Note**
>
> Você pode ignorar essa etapa se já criou sua própria política de acesso ao console do AWS Glue.

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Políticas.
3. Na lista de políticas, marque a caixa de seleção ao lado da política AWSGlueConsoleFullAccess. Você pode usar o menu Filtro e a caixa de pesquisa para filtrar a lista de políticas.
4. Selecione Ações da política e escolha Anexar.
5. Escolha o usuário ao qual a política será anexada. Você pode usar o menu Filter (Filtro) e a caixa de pesquisa para filtrar a lista de entidades principais. Depois de escolher o usuário ao qual a política será anexada, selecione Attach policy.

Anexe a política gerenciada **`AWSGlueConsoleSageMakerNotebookFullAccess`**

Você pode anexar a política `AWSGlueConsoleSageMakerNotebookFullAccess` a um usuário para gerenciar cadernos do SageMaker criados no console do AWS Glue. Além de outras permissões necessárias no console do AWS Glue, essa política concede acesso aos recursos necessários para gerenciar cadernos do SageMaker.

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Políticas.
3. Na lista de políticas, selecione a caixa de seleção ao lado de AWSGlueConsoleSageMakerNotebookFullAccess. Você pode usar o menu Filtro e a caixa de pesquisa para filtrar a lista de políticas.
4. Selecione Ações da política e escolha Anexar.
5. Escolha o usuário ao qual a política será anexada. Você pode usar o menu Filter (Filtro) e a caixa de pesquisa para filtrar a lista de entidades principais. Depois de escolher o usuário ao qual a política será anexada, selecione Attach policy.

Para anexar a política gerenciada CloudWatchLogsReadOnlyAccess

Você pode anexar a política CloudWatchLogsReadOnlyAccess a um usuário para visualizar os logs criados pelo AWS Glue no console do CloudWatch Logs.

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.

2. No painel de navegação, escolha Políticas.
3. Na lista de políticas, selecione a caixa de seleção ao lado de CloudWatchLogsReadOnlyAccess. Você pode usar o menu Filtro e a caixa de pesquisa para filtrar a lista de políticas.
4. Selecione Ações da política e escolha Anexar.
5. Escolha o usuário ao qual a política será anexada. Você pode usar o menu Filter (Filtro) e a caixa de pesquisa para filtrar a lista de entidades principais. Depois de escolher o usuário ao qual a política será anexada, selecione Attach policy.

Para anexar a política gerenciada AWSCloudFormationReadOnlyAccess

Você pode anexar a política AWSCloudFormationReadOnlyAccess a um usuário para visualizar pilhas do AWS CloudFormation usadas pelo AWS Glue no console do AWS CloudFormation.

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Políticas.
3. Na lista de políticas, marque a caixa de seleção ao lado de AWSCloudFormationReadOnlyAccess. Você pode usar o menu Filtro e a caixa de pesquisa para filtrar a lista de políticas.
4. Selecione Ações da política e escolha Anexar.
5. Escolha o usuário ao qual a política será anexada. Você pode usar o menu Filter (Filtro) e a caixa de pesquisa para filtrar a lista de entidades principais. Depois de escolher o usuário ao qual a política será anexada, selecione Attach policy.

Para anexar a política gerenciada AmazonAthenaFullAccess

Você pode anexar a política AmazonAthenaFullAccess a um usuário para visualizar dados do Amazon S3 no console do Athena.

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Políticas.
3. Na lista de políticas, marque a caixa de seleção ao lado da política AmazonAthenaFullAccess. Você pode usar o menu Filtro e a caixa de pesquisa para filtrar a lista de políticas.
4. Selecione Ações da política e escolha Anexar.

5. Escolha o usuário ao qual a política será anexada. Você pode usar o menu Filter (Filtro) e a caixa de pesquisa para filtrar a lista de entidades principais. Depois de escolher o usuário ao qual a política será anexada, selecione Attach policy.

Etapa 4: criar uma política do IAM para servidores de cadernos

Se você planeja usar notebooks com endpoints de desenvolvimento, precisará especificar permissões ao criar o servidor de notebook. Você concede essas permissões usando o AWS Identity and Access Management (IAM).

Essa política concede permissão a algumas ações do Amazon S3 para o gerenciamento dos recursos na sua conta que são exigidos pelo AWS Glue quando ele assume uma função usando essa política. Alguns dos recursos especificados nessa política referem-se a nomes padrão que são usados pelo AWS Glue para buckets do Amazon S3, scripts de ETL do Amazon S3 e recursos do Amazon EC2. Por questões de simplicidade, o AWS Glue grava por padrão alguns objetos do Amazon S3 em buckets da sua conta com o prefixo `aws-glue-*`.

> **Note**
>
> Você pode ignorar esta etapa se usar a política gerenciada pela AWS **`AWSGlueServiceNotebookRole`**.

Nesta etapa, você cria uma política semelhante a `AWSGlueServiceNotebookRole`. Você pode encontrar a versão mais atual da `AWSGlueServiceNotebookRole` no console do IAM.

Para criar uma política do IAM para cadernos

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Políticas.
3. Escolha Create Policy (Criar política).
4. Na tela Create Policy, navegue até uma guia para editar o JSON. Crie um documento de política com as seguintes instruções JSON e escolha Review policy.

```
{
   "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/*"
         ]
      }
   ]
}
```

A tabela a seguir descreve as permissões concedidas por esta política.

Ação	Recurso	Descrição
`"glue:*"`	`"*"`	Concede permissão para executar todas as operações de API do AWS Glue.
`"s3:GetBucketLocation", "s3:ListBucket", "s3:ListAllMyBuckets", "s3:GetBucketAcl"`	`"*"`	Permite listar buckets do Amazon S3 a partir de servidores de cadernos.
`"s3:GetObject"`	`"arn:aws:s3:::crawler-public*", "arn:aws:s3:::aws-glue-*"`	Permite obter objetos do Amazon S3 usados por exemplos e tutoriais dos cadernos. Convenção de nomenclatura: os nomes de buckets do Amazon S3 começam com crawler-public e aws-glue-.

Ação	Recurso	Descrição
`"s3:PutObject", "s3:DeleteObject"`	`"arn:aws:s3:::aws-glue*"`	Permite incluir e excluir objetos do Amazon S3 na sua conta a partir de cadernos. Convenção de nomenclatura: usa pastas do Amazon S3 chamadas aws-glue.
`"ec2:CreateTags", "ec2:DeleteTags"`	`"arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:instance/*"`	Permite marcar recursos do Amazon EC2 criados para servidores de cadernos. Convenção de nomenclatura: o AWS Glue marca instâncias do Amazon EC2 com aws-glue-service-resource.

5. Na tela Review Policy (Revisar política), digite o Policy Name (Nome da política), por exemplo, GlueServiceNotebookPolicyDefault. Digite uma descrição opcional e, quando estiver satisfeito com a política, escolha Create policy (Criar política).

Etapa 5: criar um perfil do IAM para servidores de cadernos

Se você planeja usar cadernos com endpoints de desenvolvimento, precisará conceder as permissões à função do IAM. Você fornece essas permissões usando o AWS Identity and Access Management IAM, por meio de uma função do IAM.

> **Note**
>
> Se você criar uma função do IAM usando o console do IAM o console criará automaticamente um perfil de instância e dará a ele o mesmo nome da função correspondente.

Para criar uma função do IAM para cadernos

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.
3. Selecione Criar função.
4. Em tipo de perfil, escolha Serviço da AWS, localize e escolha EC2, depois escolha o caso de uso EC2 e, em seguida, escolha Próximo: permissões.
5. Na página Attach permissions policy (Anexar política de permissões), escolha as políticas que contenham as permissões necessárias, por exemplo, AWSGlueServiceNotebookRole para permissões gerais do AWS Glue e a política gerenciada pela AWS AmazonS3FullAccess para acesso aos recursos do Amazon S3. Então, escolha Próximo: Análise.

> **Note**
>
> Verifique se uma das políticas nessa função concede permissões aos seus destinos e fontes do Amazon S3. Confirme também se a sua política concede acesso total ao local onde você armazena o notebook ao criar um servidor de notebook. Convém fornecer sua própria política de acesso a recursos específicos do Amazon S3. Para obter mais informações sobre como criar uma política do Amazon S3 para os seus recursos, consulte Especificar recursos em uma política.
> Se você pretende acessar as fontes e os destinos do Amazon S3 que foram criptografados com SSE-KMS, anexe uma política para permitir que os cadernos descriptografem os dados. Para obter mais informações, consulte Proteção de dados usando criptografia do lado do servidor com chaves gerenciadas pelo AWS KMS (SSE-KMS).
> Veja um exemplo a seguir.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
```

```
            ]
        }
    ]
}
```

6. Em Role name (Nome da função), digite um nome para sua função. Crie a função com um nome que tenha a string `AWSGlueServiceNotebookRole` como prefixo para permitir que a função seja transmitida dos usuários do console para o serviço de caderno. As políticas fornecidas pelo AWS Glue esperam que as funções de serviço do IAM comecem com `AWSGlueServiceNotebookRole`. Caso contrário, você precisará adicionar uma política para conceder aos seus usuários a permissão `iam:PassRole`, para que as funções do IAM correspondam às suas convenções de nomenclatura. Por exemplo, digite `AWSGlueServiceNotebookRoleDefault`. Então, escolha Criar perfil.

Etapa 6: criar uma política do IAM para cadernos do SageMaker

Ao planejar usar cadernos do SageMaker com endpoints de desenvolvimento, você deve especificar permissões ao criar o caderno. Você concede essas permissões usando o AWS Identity and Access Management (IAM).

Para criar uma política do IAM para cadernos do SageMaker

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Políticas.
3. Escolha Create Policy (Criar política).
4. Na página Create Policy (Criar política), navegue até uma guia para editar o JSON. Crie um documento de política com as seguintes instruções JSON. Edite *`bucket-name`*, *`region-code`* e *`account-id`* para o seu ambiente.

```
{
    "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/*"
            ]
        }
    ]
}
```

Em seguida, escolha Revisar política.

A tabela a seguir descreve as permissões concedidas por esta política.

Ação	Recurso	Descrição
"s3:ListBucket*"	"arn:aws:s3::: *bucket-name* "	Concede permissão para listar os buckets do Amazon S3.
"s3:GetObject"	"arn:aws:s3::: *bucket-name* *"	Concede permissão para obter objetos do Amazon S3 usados por cadernos do 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 permissão para gravar logs no Amazon CloudWatch Logs a partir de cadernos. Convenção de nomenclatura: grava grupos de logs cujos nomes começam com aws-glue.
"glue:UpdateDevEndpoint", "glue:GetDevEndpoint", "glue:GetDevEndpoints"	"arn:aws:glue: *region-code* :*account-id* :devEndpoint/*"	Concede permissão para usar um endpoint de desenvolvimento em cadernos do SageMaker.

Ação	Recurso	Descrição
`"sagemaker:ListTags"`	`"arn:aws:sagemaker:region-code:account-id:notebook-instance/*"`	Concede permissão para retornar as tags para um recurso do SageMaker. A tag `aws-glue-dev-endpoint` é obrigatória no caderno do SageMaker para conectá-lo a um endpoint de desenvolvimento.

5. Na tela Review Policy (Revisar política), insira o Policy Name (Nome da política), por exemplo, `AWSGlueSageMakerNotebook`. Digite uma descrição opcional e, quando estiver satisfeito com a política, escolha Create policy (Criar política).

Etapa 7: criar um perfil do IAM para cadernos do SageMaker

Se você planeja usar cadernos do SageMaker com endpoints de desenvolvimento, precisará conceder as permissões à função do IAM. Você fornece essas permissões usando o AWS Identity and Access Management (IAM), por meio de uma função do IAM.

Para criar uma função do IAM para cadernos do SageMaker

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.
3. Selecione Criar função.
4. Em tipo de perfil, escolha Serviço da AWS, localize e escolha SageMaker, depois escolha o caso de uso SageMaker - Execução. Então, escolha Próximo: Permissões.
5. Na página Attach permissions policy (Anexar política de permissões), escolha as políticas que contêm as permissões necessárias; por exemplo, AmazonSageMakerFullAccess. Selecione Next: Review (Próximo: revisar).

   Se você planeja acessar fontes e destinos do Amazon S3 que não foram criptografados com SSE-KMS, anexe uma política que permita que os cadernos descriptografem os dados, conforme mostrado no exemplo a seguir. Para obter mais informações, consulte Proteção de

dados usando criptografia do lado do servidor com chaves gerenciadas pelo 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. Em Role name (Nome da função), digite um nome para sua função. Para permitir que a função seja transmitida dos usuários do console para o SageMaker, use um nome com a string `AWSGlueServiceSageMakerNotebookRole` como prefixo. As políticas fornecidas pelo AWS Glue esperam que as funções do IAM comecem com `AWSGlueServiceSageMakerNotebookRole`. Caso contrário, você precisará adicionar uma política para conceder aos seus usuários a permissão `iam:PassRole`, para que as funções do IAM correspondam às suas convenções de nomenclatura.

   Por exemplom, insira `AWSGlueServiceSageMakerNotebookRole-Default` e escolha Create role (Criar função).

7. Depois de criar a função, anexe a política que concede as permissões adicionais necessárias para criar cadernos do SageMaker pelo AWS Glue.

   Abra a função que você acabou de criar, `AWSGlueServiceSageMakerNotebookRole-Default`, e escolha Attach policies (Anexar políticas). Anexe a política que você criou, denominada `AWSGlueSageMakerNotebook`, à função.

Exemplos de política de controle de acesso do AWS Glue

Esta seção contém exemplos de políticas de controle de acesso baseadas em identidades (IAM) e de políticas de recursos do AWS Glue.

Sumário

Exemplos das políticas baseadas em identidade do AWS Glue

Por padrão, os usuários e os perfis não têm permissão para criar ou modificar os recursos do AWS Glue. Eles também não podem executar tarefas usando o AWS Management Console, a AWS Command Line Interface (AWS CLI) ou a API AWS. Para conceder aos usuários permissão para executar ações nos recursos de que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis, e os usuários podem assumir os perfis.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documento de política JSON, consulte Criação de políticas do IAM no Guia do Usuário do IAM.

Para obter detalhes sobre ações e tipos de recurso definidos pelo AWS Glue, incluindo o formato dos ARNs para cada tipo de recurso, consulte Ações, recursos e chaves de condição do AWS Glue na Referência de autorização do serviço.

> **Note**
>
> Todos os exemplos fornecidos nesta seção usam a região `us-west-2`. Você pode substituí-la pela região da AWS que quiser usar.

Tópicos

- Melhores práticas de política
- As permissões em nível de recurso só se aplicam a objetos específicos do AWS Glue
- Usar o console do AWS Glue
- Permitir que os usuários visualizem suas próprias permissões
- Conceder permissões somente de leitura para uma tabela
- Filtrar tabelas pela permissão GetTables
- Conceder total acesso a uma tabela e a todas as partições
- Controlar acesso por prefixo de nome e negação explícita
- Conceder acesso usando tags
- Negar acesso usando tags
- Usar tags com as operações de API de lista e lote
- Controlar as configurações usando chaves de condição ou chaves de contexto
- Negação da capacidade de criar sessões de pré-visualização de dados para uma identidade

Melhores práticas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do AWS Glue em sua conta. Essas ações podem incorrer em custos para a Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas gerenciadas pela AWS e avance para as permissões de privilégio mínimo: para começar a conceder permissões a seus usuários e workloads, use as políticas gerenciadas pela AWS que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis na sua Conta da AWS. Recomendamos que você reduza ainda mais as permissões

definindo políticas gerenciadas pelo cliente da AWS específicas para seus casos de uso. Para mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.

- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso a ações de serviço, se elas forem usadas por meio de um AWS service (Serviço da AWS) específico, como o AWS CloudFormation. Para obter mais informações, consulte Elementos de política JSON do IAM: Condition no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações acionáveis para ajudar você a criar políticas seguras e funcionais. Para mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA): se houver um cenário que exija usuários do IAM ou um usuário raiz em sua Conta da AWS, ative a MFA para obter segurança adicional. Para exigir a MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do usuário do IAM.

Para mais informações sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

As permissões em nível de recurso só se aplicam a objetos específicos do AWS Glue

Só é possível definir um controle minucioso para objetos específicos no AWS Glue. Portanto, é necessário gravar a política do IAM do cliente para que as operações de API que permitem nomes de recurso da Amazon (ARNs) para a instrução `Resource` não sejam misturadas com operações de API que não permitem ARNs.

Por exemplo, a política do IAM a seguir permite operações de API para `GetClassifier` e `GetJobRun`. Ela define o `Resource` como `*`, pois o AWS Glue não permite ARNs para classificadores e execuções de tarefa. Como ARNs são permitidos para operações de API específicas, como `GetDatabase` e `GetTable`, os ARNs podem ser especificados na segunda metade da 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 obter uma lista de objetos do AWS Glue que permitem ARNs, consulte ARNs de recursos.

Usar o console do AWS Glue

Para acessar o console do AWS Glue, você deve ter um conjunto mínimo de permissões. Essas permissões devem autorizar você a listar e visualizar detalhes sobre os recursos do AWS Glue na sua Conta da AWS. Se você criar uma política baseada em identidade que seja mais restritiva do que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Não é necessário conceder permissões mínimas do console para usuários que fazem chamadas somente à AWS CLI ou à AWS API. Em vez disso, permita o acesso somente a ações que correspondam a operação de API que estiverem tentando executar.

Para garantir que os usuários e os perfis ainda possam usar o console do AWS Glue, anexe também a política `ConsoleAccess` ou `ReadOnly` do AWS Glue gerenciada pela AWS às entidades. Para obter mais informações, consulte Adicionando Permissões a um Usuário no Guia do Usuário do IAM.

Para que um usuário trabalhe com o console do AWS Glue, ele deve ter um conjunto mínimo de permissões que liberem o trabalho com os recursos do AWS Glue na conta da AWS. Além dessas permissões do AWS Glue, o console exige permissões dos seguintes serviços:

- Permissões do Amazon CloudWatch Logs para exibir logs.
- Permissões do AWS Identity and Access Management (IAM) para listar e transmitir funções.
- Permissões do AWS CloudFormation para trabalhar com pilhas.
- Permissões do Amazon Elastic Compute Cloud (Amazon EC2) para listar VPCs, sub-redes, grupos de segurança, instâncias e outros objetos.
- Permissões do Amazon Simple Storage Service (Amazon S3) para listar buckets e objetos e para recuperar e salvar scripts.
- Permissões do Amazon Redshift necessárias para trabalhar com clusters.
- Permissões do Amazon Relational Database Service (Amazon RDS) para listar instâncias.

Para obter mais informações sobre as permissões que os usuários necessitam para visualizar e trabalhar com o console do AWS Glue, consulte Etapa 3: anexar uma política aos usuários ou grupos que acessam o AWS Glue.

Se você criar uma política do IAM que seja mais restritiva que as permissões mínimas necessárias, o console do não funcionará como pretendido para os usuários com essa política do IAM. Para garantir que esses usuários ainda possam usar o console do AWS Glue, anexe também a política gerenciada `AWSGlueConsoleFullAccess`, conforme descrito em AWS políticas gerenciadas (predefinidas) para AWS Glue.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como você pode criar uma política que permite que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política

inclui permissões para concluir essa ação no console ou de forma programática usando a AWS CLI ou a AWS API.

```
{
    "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 permissões somente de leitura para uma tabela

A política a seguir concede permissão somente leitura a uma tabela `books` no banco de dados db1. Para obter mais informações sobre como usar nomes de recurso da Amazon (ARNs), consulte ARNs do 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"
      ]
    }
  ]
}
```

Essa política concede permissão somente leitura a uma tabela chamada `books` no banco de dados chamado `db1`. Para conceder a permissão `Get` para uma tabela, também é necessário conceder permissão para os recursos do catálogo e do banco de dados.

A política a seguir concede as permissões mínimas necessárias para um criar a tabela `tb1` no banco de dados `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 tabelas pela permissão GetTables

Suponha que há três tabelas: `customers`, `stores` e `store_sales` no banco de dados `db1`. A política a seguir concede a permissão `GetTables` a `stores` e `store_sales`, mas não a `customers`. Quando você chama `GetTables` com essa política, o resultado contém somente as duas tabelas autorizadas (a tabela `customers` não é retornada).

```
{
    "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"
            ]
        }
    ]
}
```

Você pode simplificar a política anterior usando `store*` para corresponder a todos os nomes de tabelas que comecem com `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*"
```

```
            ]
        }
    ]
}
```

Da mesma forma, usando `/db1/*` para corresponder a todas as tabelas no `db1`, a política a seguir concede o acesso `GetTables` a todas as tabelas no `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/*"
            ]
        }
    ]
}
```

Se nenhum ARN de tabela for fornecido, uma chamada para `GetTables` será bem-sucedida, mas retornará uma lista vazia.

```
{
    "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"
            ]
```

```
        }
    ]
}
```

Se o ARN do banco de dados estiver ausente na política, haverá uma falha na chamada para `GetTables` com uma `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 total acesso a uma tabela e a todas as partições

A política a seguir concede todas as permissões em uma tabela denominada `books` no banco de dados `db1`. Isso inclui permissões de leitura e gravação na própria tabela, em versões arquivadas da tabela e em todas as suas partições.

```
{
    "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"
                        ]
        }
    ]
}
```

A política anterior pode ser simplificada na prática.

```
{
    "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"
            ]
        }
    ]
}
```

Observe que a granularidade mínima de um controle de acesso granular é no nível de tabela. Isso significa que você não pode conceder a um usuário acesso a algumas partições em uma tabela, mas não a outras, ou a algumas colunas da tabela, mas não a outras. Um usuário tem acesso a tudo ou a nada de uma tabela.

Controlar acesso por prefixo de nome e negação explícita

Neste exemplo, suponha que os bancos de dados e tabelas em seu AWS Glue Data Catalog são organizados usando prefixos de nomes. Os bancos de dados no estágio de desenvolvimento têm o prefixo de nome `dev-`, e os em produção têm o prefixo de nome `prod-`. Você pode usar a política a seguir para conceder aos desenvolvedores acesso total a todos os bancos de dados, tabelas, UDFs, e assim por diante, que têm o prefixo `dev-`. Mas você concede acesso somente leitura a tudo com o prefixo `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-*"
            ]
        }
    ]
}
```

A segunda instrução na política anterior usa `deny` explícita. Você pode usar `deny` explícita para substituir todas as permissões `allow` que foram concedidas à entidade principal. Isso permite bloquear o acesso a recursos críticos e impedir que outra política conceda acesso a eles acidentalmente.

No exemplo anterior, embora a primeira instrução conceda acesso total aos recursos `prod-`, a segunda instrução revoga explicitamente o acesso de gravação neles, deixando apenas o acesso de leitura aos recursos `prod-`.

Conceder acesso usando tags

Por exemplo, suponha que você deseja limitar o acesso a um gatilho `t2` para um usuário específico chamado `Tom` em sua conta. Todos os outros usuários, inclusive `Sam`, têm acesso ao gatilho `t1`. Os gatilhos `t1` e `t2` têm as seguintes propriedades.

```
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 * * ? *)"
        }
    ]
}
```

O administrador do AWS Glue anexou um valor de tag `Tom` (`aws:ResourceTag/Name": "Tom"`) ao gatilho `t2`. O administrador do AWS Glue também concedeu a Tom uma política do IAM com uma instrução de condição baseada na tag. Como resultado, Tom só pode usar uma operação do AWS Glue que atue sobre recursos com o valor de tag `Tom`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
         "Effect": "Allow",
         "Action": "glue:*",
         "Resource": "*",
         "Condition": {
             "StringEquals": {
                "aws:ResourceTag/Name": "Tom"
              }
          }
         }
    ]
}
```

Quando Tom tentar acessar o gatilho `t1`, ele receberá uma mensagem de acesso negado. Enquanto isso, ele pode recuperar o gatilho `t2` com êxito.

```
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 não pode usar a operação de API `GetTriggers` plural para listar triggers porque essa operação não é compatível com filtragem por tags.

Para conceder acesso a `GetTriggers` para Tom, o administrador do AWS Glue cria uma política que divide as permissões em duas seções. Uma seção permite que Tom acesse todos os gatilhos com a operação da API `GetTriggers`. A segunda seção permite que Tom acesse as operações de API marcadas com o valor `Tom`. Com essa política, Tom tem permissão de acesso a `GetTriggers` e `GetTrigger` para o gatilho `t2`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "glue:GetTriggers",
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "glue:*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Name": "Tom"
                }
            }
        }
    ]
}
```

Negar acesso usando tags

Outra abordagem de uma política de recurso é negar explicitamente o acesso aos recursos.

> **Important**
>
> Uma política de negação explícita não funciona para operações de API no plural, como `GetTriggers`.

No exemplo de política a seguir, todas as operações de trabalho do AWS Glue são permitidas. Porém, a segunda instrução `Effect` nega explicitamente acesso a trabalhos marcados com a chave `Team` e o valor `Special`.

Quando um administrador anexa a política a seguir a uma identidade, essa entidade pode acessar todos os trabalhos, exceto os marcados com a chave `Team` e o 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 tags com as operações de API de lista e lote

Uma terceira abordagem para escrever uma política de recurso é permitir o acesso aos recursos usando uma operação da API `List` para listar os recursos para um valor de tag. Depois, use a operação da API `Batch` correspondente para permitir o acesso aos detalhes de recursos específicos. Com essa abordagem, o administrador não precisa conceder acesso às operações de API plurais `GetCrawlers`, `GetDevEndpoints`, `GetJobs` ou `GetTriggers`. Em vez disso, você pode permitir a possibilidade de listar os recursos com as seguintes operações de API:

- `ListCrawlers`
- `ListDevEndpoints`
- `ListJobs`
- `ListTriggers`

E você pode permitir a capacidade de obter detalhes sobre os recursos individuais com as seguintes operações de API:

- `BatchGetCrawlers`
- `BatchGetDevEndpoints`
- `BatchGetJobs`
- `BatchGetTriggers`

Como administrador, para usar essa abordagem, faça o seguinte:

1. Adicionar tags a seus crawlers, endpoints de desenvolvimento, trabalhos e gatilhos.
2. Negar o acesso de usuário a operações de API `Get`, como `GetCrawlers`, `GetDevEndponts`, `GetJobs` e `GetTriggers`.

3. Para permitir que os usuários descubram a quais recursos marcados eles têm acesso, conceda acesso de usuário a operações de API `List`, como `ListCrawlers`, `ListDevEndponts`, `ListJobs` e `ListTriggers`.
4. Negar o acesso de usuário a APIs de marcação do AWS Glue, como `TagResource` e `UntagResource`.
5. Conceder acesso de usuário aos detalhes de recursos com operações de API `BatchGet`, como `BatchGetCrawlers`, `BatchGetDevEndponts`, `BatchGetJobs` e `BatchGetTriggers`.

Por exemplo, ao chamar a operação `ListCrawlers`, forneça um valor de tag de acordo com o nome do usuário. O resultado será uma lista de crawlers que correspondem aos valores de tag fornecidos. Forneça a lista de nomes ao `BatchGetCrawlers` para obter detalhes sobre cada crawler com a tag dada.

Por exemplo, se Tom só deverá ser capaz de recuperar detalhes de triggers marcados com `Tom`, o administrador poderá adicionar tags aos triggers para `Tom`, negar o acesso à operação de API `GetTriggers` a todos os usuários e permitir o acesso de todos os usuários a `ListTriggers` e `BatchGetTriggers`.

Esta é a política de recurso que o administrador do AWS Glue concede a Tom. Na primeira seção da política, as operações de API do AWS Glue são negadas para `GetTriggers`. Na segunda seção da política, `ListTriggers` é permitida para todos os recursos. No entanto, na terceira seção, esses recursos marcados com `Tom` têm o acesso permitido com o acesso a `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"
                }
            }
        }
    ]
}
```

Usando os mesmos gatilhos do exemplo anterior, Tom pode acessar o gatilho `t2`, mas não o gatilho `t1`. O exemplo a seguir mostra os resultados quando Tom tenta acessar `t1` e `t2` com `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.
```

O exemplo a seguir mostra os resultados quando Tom tenta acessar ambos os gatilhos `t2` e `t3` (que não existe) na mesma chamada de `BatchGetTriggers`. Observe que, como Tom tem acesso ao gatilho `t2` e ele existe, somente `t2` é retornado. Embora Tom tenha permissão para acessar o gatilho `t3`, `t3` não existe, de modo que `t3` é retornado na resposta em uma 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 as configurações usando chaves de condição ou chaves de contexto

Você pode usar chaves de condição ou chaves de contexto ao conceder permissões para criar e atualizar trabalhos. Estas seções discutem as chaves:

- Controlar políticas que controlam configurações usando chaves de condição
- Controlar políticas que controlam configurações usando chaves de contexto

Controlar políticas que controlam configurações usando chaves de condição

O AWS Glue fornece três chaves de condição do IAM, `glue:VpcIds`, `glue:SubnetIds` e `glue:SecurityGroupIds`. Você pode usar chaves de condição nas políticas do IAM ao conceder permissões para criar e atualizar trabalhos. Você pode usar essa configuração para garantir que os trabalhos ou sessões não sejam criados (ou atualizados) para serem executados fora de um ambiente de VPC desejado. As informações de configuração da VPC não são uma entrada direta da solicitação `CreateJob`, mas são inferidas do campo "conexões" do trabalho que aponta para uma conexão do AWS Glue.

Exemplo de uso

Crie uma conexão de tipo de rede do AWS Glue denominada "traffic-monitored-connection" com o VpcId "vpc-id1234", SubnetIds e SecurityGroupIds desejados.

Especifique a condição das chaves de condição para as ações `CreateJob` e `UpdateJob` na política do IAM.

```
{
  "Effect": "Allow",
  "Action": [
    "glue:CreateJob",
    "glue:UpdateJob"
  ],
  "Resource": [
    "*"
  ],
  "Condition": {
    "ForAnyValue:StringLike": {
      "glue:VpcIds": [
        "vpc-id1234"
      ]
    }
  }
}
```

Você pode criar uma política do IAM semelhante para proibir a criação de um trabalho do AWS Glue sem especificar informações de conexão.

Restringir sessões em VPCs

Para forçar que as sessões criadas sejam executadas em uma VPC especificada, você restringe a permissão do perfil adicionando um efeito `Deny` na ação `glue:CreateSession` com a condição de que glue:vpc-id not equal to vpc-<123>. Por exemplo:

```
"Effect": "Deny",
"Action": [
    "glue:CreateSession"
 ],
"Condition": {
    "StringNotEquals" : {"glue:VpcIds" : ["vpc-123"]}
}
```

Você também pode forçar que as sessões criadas sejam executadas em uma VPC adicionando um efeito `Deny` na ação `glue:CreateSession` com a condição de que `glue:vpc-id` seja nulo. Por exemplo:

```
{
    "Effect": "Deny",
    "Action": [
        "glue:CreateSession"
    ],
      "Condition": {
        "Null": {"glue:VpcIds": true}
    }
},
{
    "Effect": "Allow",
    "Action": [
        "glue:CreateSession"
    ],
    "Resource": ["*"]
}
```

Controlar políticas que controlam configurações usando chaves de contexto

O Glue da AWS fornece uma chave de contexto (`glue:CredentialIssuingService=glue.amazonaws.com`) para cada sessão de função que o AWS Glue disponibiliza para o endpoint do trabalho e do desenvolvedor. Isso permite que você implemente controles de segurança para as medidas tomadas pelo scripts do AWS Glue. O AWS Glue fornece outra chave de contexto (`glue:RoleAssumedBy=glue.amazonaws.com`) para cada sessão de perfil em que o AWS Glue faz uma chamada para outro serviço da AWS em nome do cliente (não por um endpoint de trabalho/desenvolvimento, mas diretamente pelo serviço AWS Glue).

Exemplo de uso

Especifique a permissão condicional na política do IAM e anexe-a ao perfil a ser usado por um trabalho do AWS Glue. Isso garante que determinadas ações sejam permitidas/negadas dependendo de a sessão de perfil ser ou não usada para um ambiente de runtime de trabalho do AWS Glue.

```
{
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::confidential-bucket/*",
    "Condition": {
        "StringEquals": {
            "glue:CredentialIssuingService": "glue.amazonaws.com"
        }
    }
}
```

Negação da capacidade de criar sessões de pré-visualização de dados para uma identidade

Essa seção contém um exemplo de política do IAM usado para negar a capacidade de criar sessões de pré-visualização de dados para uma identidade. Anexe essa política à identidade, que é separada do perfil usado pela sessão de pré-visualização de dados durante sua execução.

```
{
    "Sid": "DatapreviewDeny",
    "Effect": "Deny",
     "Action": [
           "glue:CreateSession"
     ],
     "Resource": [
          "arn:aws:glue:*:*:session/glue-studio-datapreview*"
      ]
 }
```

Exemplos de política baseada em recursos para o AWS Glue

Esta seção contém exemplos de políticas baseadas em recursos, incluindo políticas que concedem acesso entre contas.

Os exemplos a seguir usam a AWS Command Line Interface (AWS CLI) para interagir com as operações de API do serviço AWS Glue. Você pode executar as mesmas operações no console do AWS Glue ou usar um dos SDKs da AWS.

> **Important**
>
> Ao alterar uma política de recurso do AWS Glue, você pode acidentalmente revogar permissões para usuários do AWS Glue existentes em sua conta e provocar interrupções

inesperadas. Antes de fazer as alterações, tente estes exemplos apenas em contas de desenvolvimento ou de teste e garanta que elas não interrompam nenhum fluxo de trabalho existente.

Tópicos

Considerações sobre o uso de políticas baseadas em recurso com o AWS Glue

> **Note**
>
> As políticas do IAM e uma política de recursos do AWS Glue levam alguns segundos para serem propagadas. Depois de anexar uma nova política, você pode perceber que a política antiga ainda está em vigor até que a nova política tenha sido propagada pelo sistema.

Você usa um documento de política escrito em formato JSON para criar ou modificar uma política de recurso. A sintaxe da política é a mesma que a de uma política do IAM baseada em identidade (consulte Referência de política JSON do IAM), com as seguintes exceções:

- Um bloco `"Principal"` ou `"NotPrincipal"` é necessário para cada declaração da política.
- O `"Principal"` ou o `"NotPrincipal"` devem identificar as entidades principais válidos existentes. Padrões de curinga (como `arn:aws:iam::`*`account-id`*`:user/*`) não são permitidos.
- O bloco `"Resource"` da política exige que todos os ARNs de recursos correspondam à seguinte sintaxe de expressão regular (em que o primeiro `%s` é a *`região`* e o segundo `%s` é o *`account-id`*):

```
*arn:aws:glue:%s:%s:(\*|[a-zA-Z\*]+\/?.*)
```

  Por exemplo, `arn:aws:glue:us-west-2:`*`account-id`*`:*` e `arn:aws:glue:us-west-2:`*`account-id`*`:database/default` são permitidos, mas `*` não é permitido.

- Ao contrário das políticas baseadas em identidade, a política de recurso do AWS Glue deve conter apenas os nomes dos recursos da Amazon (ARN) que pertencem ao catálogo ao qual a política está anexada. Esses ARNs sempre começam com `arn:aws:glue:`.
- Uma política não pode impedir que a identidade criada por ela crie ou modifique mais a política.
- Um documento JSON de política de recurso não pode exceder 10 KB.

Usar uma política de recurso para controlar o acesso na mesma conta

Neste exemplo, um usuário administrador na conta A cria uma política de recursos que concede à usuária do IAM `Alice` acesso total ao catálogo na conta A. Alice não tem nenhuma política do IAM anexada.

Para fazer isso, o usuário administrador executa o comando da AWS CLI a seguir.

```
# 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:*"
      ]
    }
  ]
}'
```

Em vez de inserir o documento de política JSON como parte do comando da AWS CLI, você pode salvar um documento de política em um arquivo e fazer referência ao caminho do arquivo no comando da AWS CLI, com o prefixo `file://`. Este é um exemplo de como é possível fazer isso:

```
$ 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
```

Depois que essa política de recurso tiver sido propagada, Alice poderá acessar todos os recursos do AWS Glue na Conta A, conforme a seguir.

```
# 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"}
```

Em resposta à chamada `get-table` de Alice, o serviço AWS Glue retorna o seguinte:

```
{
  "Table": {
    "Name": "tbl1",
    "PartitionKeys": [],
    "StorageDescriptor": {
        ......
```

```
    },
    ......
  }
}
```

AWS políticas gerenciadas para AWS Glue

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas operações de API são disponibilizadas para serviços existentes.

Para mais informações, consulte Políticas gerenciadas pela AWS no Manual do usuário do IAM.

AWS políticas gerenciadas (predefinidas) para AWS Glue

AWS aborda muitos casos de uso comuns fornecendo políticas autônomas do IAM que são criadas e administradas pela AWS. Essas políticas AWS gerenciadas concedem as permissões necessárias para casos de uso comuns, para que você possa evitar a necessidade de investigar quais permissões são necessárias. Para mais informações, consulte Políticas gerenciadas pela AWS no Manual do usuário do IAM.

As seguintes políticas AWS gerenciadas, que você pode anexar às identidades em sua conta, são específicas AWS Glue e estão agrupadas por cenário de caso de uso:

- AWSGlueConsoleFullAccess— Concede acesso total aos AWS Glue recursos quando uma identidade à qual a política está anexada usa AWS Management Console o. Se você seguir a convenção de nomenclatura para os recursos especificados nesta política, os usuários poderão acessar todos os recursos do console. Esta política geralmente é anexada aos usuários do console do AWS Glue.

- AWSGlueServiceRole— Concede acesso aos recursos que vários AWS Glue processos exigem para serem executados em seu nome. Esses recursos incluem AWS Glue Amazon S3, IAM, CloudWatch Logs e Amazon EC2. Se você seguir a convenção de nomenclatura para os recursos especificados nesta política, os processos do AWS Glue terão as permissões necessárias. Esta política geralmente é anexada a funções especificadas durante a definição de crawlers, trabalhos e endpoints de desenvolvimento.
- AwsGlueSessionUserRestrictedServiceRole— Fornece acesso total a todos os AWS Glue recursos, exceto às sessões. Permite que os usuários criem e usem somente as sessões interativas associadas ao usuário. Essa política inclui outras permissões necessárias AWS Glue para gerenciar AWS Glue recursos em outros AWS serviços. A política também permite adicionar tags a AWS Glue recursos em outros AWS serviços.

  > **Note**
  >
  > Para obter todos os benefícios de segurança, não conceda esta política a um usuário que tenha atribuída a política `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` ou `AWSGlueConsoleSageMakerNotebookFullAccess`.

- AwsGlueSessionUserRestrictedPolicy— Fornece acesso para criar sessões AWS Glue interativas usando a operação da `CreateSession` API somente se uma chave de tag “proprietário” e um valor que correspondam ao ID de AWS usuário do destinatário forem fornecidos. Essa política de identidade está anexada ao usuário do IAM que invoca a operação de API `CreateSession`. Essa política também permite que o destinatário interaja com os recursos da sessão AWS Glue interativa que foram criados com uma tag de “proprietário” e um valor que correspondam à sua ID de usuário. AWS Essa política nega permissão para alterar ou remover tags de "proprietário" de um recurso de sessão do AWS Glue após a criação da sessão.

  > **Note**
  >
  > Para obter todos os benefícios de segurança, não conceda essa política a um usuário que tenha atribuída a política `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` ou `AWSGlueConsoleSageMakerNotebookFullAccess`.

- AwsGlueSessionUserRestrictedNotebookServiceRole— Fornece acesso suficiente à sessão do AWS Glue Studio notebook para interagir com recursos específicos da sessão AWS Glue interativa. Esses são recursos criados com o valor da tag “owner” que corresponde ao ID de AWS usuário do principal (usuário ou função do IAM) que cria o notebook. Para obter mais informações

sobre essas tags, consulte o gráfico Valores de chave da entidade principal no Guia do usuário do IAM.

Essa política de perfil de serviço é anexada ao perfil especificado com um comando magic no caderno ou é passado como um perfil para a operação de API `CreateSession`. Essa política também permite que o diretor crie uma sessão AWS Glue interativa a partir da interface do AWS Glue Studio notebook somente se a tag, a chave “proprietário” e o valor corresponderem ao ID de AWS usuário do principal. Essa política nega permissão para alterar ou remover tags de "proprietário" de um recurso de sessão do AWS Glue após a criação da sessão. Essa política também inclui permissões para gravação e leitura de buckets do Amazon S3, gravação de CloudWatch registros e criação e exclusão de tags para recursos do Amazon EC2 usados pelo. AWS Glue

> **Note**
>
> Para obter todos os benefícios de segurança, não conceda essa política a uma função que tenha atribuída a política `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` ou `AWSGlueConsoleSageMakerNotebookFullAccess`.

- AwsGlueSessionUserRestrictedNotebookPolicy— Fornece acesso para criar uma sessão AWS Glue interativa a partir da interface do AWS Glue Studio notebook somente se houver uma tag, chave “proprietário” e um valor que correspondam à ID do AWS usuário principal (usuário ou função do IAM) que cria o notebook. Para obter mais informações sobre essas tags, consulte o gráfico Valores de chave da entidade principal no Guia do usuário do IAM.

  Essa política é anexada à entidade principal (usuário ou perfil do IAM) que cria sessões na interface do caderno do AWS Glue Studio. Essa política também permite acesso suficiente ao caderno do AWS Glue Studio para interação com recursos específicos de sessão interativa do AWS Glue. Esses são recursos criados com o valor da tag “owner” que corresponde ao ID de AWS usuário do principal. Essa política nega permissão para alterar ou remover tags de "proprietário" de um recurso de sessão do AWS Glue após a criação da sessão.
- AWSGlueServiceNotebookRole— Concede acesso às AWS Glue sessões iniciadas em um AWS Glue Studio caderno. Essa política permite listar e obter informações de sessão para todas as sessões, mas só permite que os usuários criem e usem as sessões marcadas com sua ID de AWS usuário. Essa política nega permissão para alterar ou remover tags de “proprietário” dos recursos da AWS Glue sessão marcados com seu AWS ID.

Atribua essa política ao AWS usuário que cria trabalhos usando a interface do notebook noAWS Glue Studio.

- AWSGlueConsoleSageMakerNotebookFullAccess— Concede acesso total ao AWS Glue e aos SageMaker recursos quando a identidade à qual a política está anexada usa AWS Management Console o. Se você seguir a convenção de nomenclatura para os recursos especificados nesta política, os usuários poderão acessar todos os recursos do console. Essa política geralmente é anexada aos usuários do AWS Glue console que gerenciam SageMaker notebooks.
- AWSGlueSchemaRegistryFullAccess— Concede acesso total aos recursos do AWS Glue Schema Registry quando a identidade à qual a política está anexada usa o AWS Management Console ou AWS CLI. Se você seguir a convenção de nomenclatura para os recursos especificados nesta política, os usuários poderão acessar todos os recursos do console. Essa política geralmente é anexada aos usuários do AWS Glue console ou AWS CLI que gerenciam o Registro do AWS Glue Esquema.
- AWSGlueSchemaRegistryReadonlyAccess— Concede acesso somente para leitura aos recursos do AWS Glue Schema Registry quando uma identidade à qual a política está anexada usa o ou. AWS Management Console AWS CLI Se você seguir a convenção de nomenclatura para os recursos especificados nesta política, os usuários poderão acessar todos os recursos do console. Essa política geralmente é anexada aos usuários do AWS Glue console ou AWS CLI que usam o Registro do AWS Glue Esquema.

> **Note**
>
> É possível analisar essas políticas de permissões fazendo login no console do IAM e pesquisando políticas específicas.

Você também pode criar suas próprias políticas do IAM personalizadas a fim de conceder permissões para ações e recursos do AWS Glue. Você pode anexar essas políticas personalizadas a usuários ou grupos do IAM que exijam essas permissões.

AWS Glue atualizações em políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do AWS Glue desde que esse serviço começou a monitorar essas alterações. Para receber alertas automáticos sobre alterações nesta página, assine o feed RSS na página de histórico do AWS Glue Document.

Alteração	Descrição	Data
AwsGlueSessionUserRestrictedPolicy — Pequena atualização em uma política existente.	Adicione `glue:StartCompletion` e `glue:GetCompletion` à política. Necessário para a integração de dados do Amazon Q no AWS Glue.	30 de abril de 2024
AwsGlueSessionUserRestrictedNotebookServiceRole — Pequena atualização em uma política existente.	Adicione `glue:StartCompletion` e `glue:GetCompletion` à política. Necessário para a integração de dados do Amazon Q no AWS Glue.	30 de abril de 2024
AwsGlueSessionUserRestrictedServiceRole — Pequena atualização em uma política existente.	Adicione `glue:StartCompletion` e `glue:GetCompletion` à política. Necessário para a integração de dados do Amazon Q no AWS Glue.	30 de abril de 2024
AWSGlueServiceNotebookRole — Pequena atualização em uma política existente.	Adicione `glue:StartCompletion` e `glue:GetCompletion` à política. Necessário para a integração de dados do Amazon Q no AWS Glue.	30 de janeiro de 2024
AwsGlueSessionUserRestrictedNotebookPolicy — Pequena atualização em uma política existente.	Adicione `glue:StartCompletion` e `glue:GetCompletion` à política. Necessário para a integração de dados do Amazon Q no AWS Glue.	29 de novembro de 2023

Alteração	Descrição	Data
AWSGlueServiceNotebookRole — Pequena atualização em uma política existente.	Adicionar `codewhisperer:GenerateRecommendations` à política. Necessário para um novo recurso em que o AWS Glue gera CodeWhisperer recomendações.	9 de outubro de 2023
AWSGlueServiceRole — Pequena atualização em uma política existente.	Aumente o escopo das CloudWatch permissões para refletir melhor o registro do AWS Glue.	4 de agosto de 2023
AWSGlueConsoleFullAccess — Pequena atualização em uma política existente.	Adicione uma lista de fórmulas de `databrew` e descreva as permissões à política. Necessário para fornecer acesso administrativo total aos novos recursos em que o AWS Glue possa acessar receitas.	9 de maio de 2023
AWSGlueConsoleFullAccess — Pequena atualização em uma política existente.	Adicionar `cloudformation:ListStacks` à política. Preserva os recursos existentes após alterações nos requisitos de AWS CloudFormation autorização.	28 de março de 2023

Alteração	Descrição	Data
Adicionadas novas políticas gerenciadas para o recurso de sessões interativas: • AwsGlueSessionUser RestrictedServiceRole • AwsGlueSessionUser RestrictedPolicy • AwsGlueSessionUser RestrictedNotebook ServiceRole • AwsGlueSessionUser RestrictedNotebookPolicy	Essas políticas visam fornecer segurança adicional para sessões interativas e cadernos no AWS Glue Studio. As políticas restringe m o acesso à operação de API `CreateSession` para que somente o proprietário tenha acesso.	30 de novembro de 2021
AWSGlueConsoleSage MakerNotebookFullAccess — Atualização de uma política existente.	Removido um ARN de recurso redundante (`arn:aws:s 3:::aws-glue-*/*`) para a ação que concede permissões de leitura/g ravação nos buckets do Amazon S3 que o AWS Glue usa para armazenar scripts e arquivos temporários. Corrigido um problema de sintaxe alterando `"StringEq uals"` para `"ForAnyVa lue:StringLike"` , e movidas as linhas `"Effect": "Allow"` para precederem oa linha `"Action":` em cada lugar em que elas estavam fora de ordem.	15 de julho de 2021

Alteração	Descrição	Data
AWSGlueConsoleFullAccess — Atualização de uma política existente.	Removido um ARN de recurso redundante (`arn:aws:s3:::aws-glue-*/*`) para a ação que concede permissões de leitura/gravação nos buckets do Amazon S3 que o AWS Glue usa para armazenar scripts e arquivos temporários.	15 de julho de 2021
O AWS Glue começo a rastrear alterações.	AWS Gluecomeçou a rastrear as mudanças em suas políticas AWS gerenciadas.	10 de junho de 2021

Especificação de ARNs de recursos do AWS Glue

No AWS Glue, você pode controlar o acesso a recursos usando uma política do AWS Identity and Access Management (IAM). Em uma política, você usa um Nome de recurso da Amazon (ARN) para identificar o recurso a que a política se aplica. Nem todos os recursos do AWS Glue oferecem suporte a ARNs.

Tópicos

ARNs do Data Catalog

Os recursos do Data Catalog têm uma estrutura hierárquica, com `catalog` como a raiz.

```
arn:aws:glue:region:account-id:catalog
```

Cada conta da AWS tem um único Data Catalog em uma região da AWS com o ID da conta de 12 dígitos como o ID do catálogo. Os recursos têm ARNs exclusivos associados a eles, como exibido na tabela a seguir.

Tipo de atributo	Formato ARN
Catálogo	`arn:aws:glue:` *`region`*`:`*`account-id`* `:catalog` Por exemplo: `arn:aws:glue:us-east-1:123456789012:catalog`
Banco de dados	`arn:aws:glue:` *`region`*`:`*`account-id`* `:database/` *`database name`* Por exemplo: `arn:aws:glue:us-east-1:123456789012:database/db1`
Tabela	`arn:aws:glue:` *`region`*`:`*`account-id`* `:table/`*`database name`*`/`*`table name`* Por exemplo: `arn:aws:glue:us-east-1:123456789012:table/db1/tbl1`
Função definida pelo usuário	`arn:aws:glue:` *`region`*`:`*`account-id`* `:userDefinedFunction/` *`database name`*`/`*`user-defined function name`* Por exemplo: `arn:aws:glue:us-east-1:123456789012:userDefinedFunction/db1/func1`
Conexão	`arn:aws:glue:` *`region`*`:`*`account-id`* `:connection/` *`connection name`* Por exemplo: `arn:aws:glue:us-east-1:123456789012:connection/connection1`
Sessão interativa	`arn:aws:glue:` *`region`*`:`*`account-id`* `:session/` *`interactive session id`*

Tipo de atributo	Formato ARN
	Por exemplo: `arn:aws:glue:us-east-1:123456789012:session/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`

Para habilitar um controle de acesso minucioso, você pode usar esses ARNs em suas políticas do IAM e políticas de recursos para conceder e negar acesso a recursos específicos. Curingas são permitidos nas políticas. Por exemplo, o seguinte ARN corresponde a todas as tabelas no banco de dados `default`.

```
arn:aws:glue:us-east-1:123456789012:table/default/*
```

Important

Todas as operações executadas em um recurso do Data Catalog exigem permissão no recurso e em todos os ancestrais desse recurso. Por exemplo, criar uma partição para uma tabela requer permissão na tabela, no banco de dados e no catálogo em que a tabela está localizada. O exemplo a seguir mostra a permissão necessária para criar partições na tabela `PrivateTable` do banco de dados `PrivateDatabase` no 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"
   ]
}
```

Além de permissão no recurso e em todos os seus ancestrais, todas as operações de exclusão exigem permissão em todos os filhos desse recurso. Por exemplo, excluir um banco de dados requer permissão em todas as tabelas e funções definidas pelo usuário no banco de dados, além do banco de dados e no catálogo em que o banco de dados está

localizado. O exemplo a seguir mostra a permissão necessária para excluir o banco de dados `PrivateDatabase` no 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"
   ]
}
```

Resumindo, as ações em recursos do Data Catalog seguem estas regras de permissão:

- As ações no catálogo requerem permissão somente no catálogo.
- As ações em um banco de dados requerem permissão no banco de dados e no catálogo.
- A exclusão de ações em um banco de dados requer permissão no banco de dados e no catálogo e em todas as tabelas e funções definidas pelo usuário no banco de dados.
- As ações em uma tabela, partição ou versão de tabela requerem permissão na tabela, no banco de dados e no catálogo.
- As ações em uma função definida pelo usuário requerem permissão na função definida pelo usuário, no banco de dados e no catálogo.
- As ações em uma conexão requerem permissão na conexão e no catálogo.

ARNs de objetos fora do catálogo no AWS Glue

Alguns recursos do AWS Glue permitem permissões no nível do recurso para controlar o acesso usando um ARN. Você pode usar esses ARNs em suas políticas do IAM para permitir um controle de acesso refinado. A tabela a seguir lista os recursos que podem conter ARNs de recursos.

Tipo de atributo	Formato ARN
Crawler	`arn:aws:glue:` *`region`*`:`*`account-id`* `:crawler/` *`crawler-name`* Por exemplo: `arn:aws:glue:us-east-1:123456789012:crawler/mycrawler`
Trabalho	`arn:aws:glue:` *`region`*`:`*`account-id`* `:job/`*`job-name`* Por exemplo: `arn:aws:glue:us-east-1:123456789012:job/testjob`
Trigger	`arn:aws:glue:` *`region`*`:`*`account-id`* `:trigger/` *`trigger-name`* Por exemplo: `arn:aws:glue:us-east-1:123456789012:trigger/sampletrigger`
Endpoint de desenvolvimento	`arn:aws:glue:` *`region`*`:`*`account-id`* `:devEndpoint/` *`development-endpoint-name`* Por exemplo: `arn:aws:glue:us-east-1:123456789012:devEndpoint/temporarydevendpoint`
Transformação de machine learning	`arn:aws:glue:` *`region`*`:`*`account-id`* `:mlTransform/` *`transform-id`* Por exemplo: `arn:aws:glue:us-east-1:123456789012:mlTransform/tfm-1234567890`

Controle de acesso para operações singulares da API fora do catálogo do AWS Glue

As operações singulares de API fora do catálogo do AWS Glue atuam em um único item (endpoint de desenvolvimento). Os exemplos são `GetDevEndpoint`, `CreateUpdateDevEndpoint` e `UpdateDevEndpoint`. Para essas operações, uma política deve colocar o nome da API no bloco `"action"` e o ARN do recurso no bloco `"resource"`.

Suponha que você queira permitir que um usuário chame a operação `GetDevEndpoint`. A política a seguir concede as permissões mínimas necessárias para um endpoint chamado `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"
        }
    ]
}
```

A política a seguir permite o acesso `UpdateDevEndpoint` a recursos que correspondem a `myDevEndpoint-` com um caractere curinga (*).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionWithWildcard",
            "Effect": "Allow",
            "Action": "glue:UpdateDevEndpoint",
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/myDevEndpoint-
*"
        }
    ]
}
```

Você pode combinar as duas políticas, como no exemplo a seguir. Você pode ver `EntityNotFoundException` para qualquer endpoint de desenvolvimento cujo nome começa com `A`. No entanto, um erro de acesso negado retorna quando você tenta acessar outros endpoints de desenvolvimento.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "CombinedPermissions",
            "Effect": "Allow",
            "Action": [
                "glue:UpdateDevEndpoint",
                "glue:GetDevEndpoint"
            ],
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/A*"
        }
    ]
}
```

Controle de acesso para operações da API fora do catálogo do AWS Glue que recuperam vários itens

Algumas operações da API do AWS Glue recuperam vários itens (como vários endpoints de desenvolvimento); por exemplo, `GetDevEndpoints`. Para essa operação, você pode especificar apenas um recurso de caractere curinga (*), e não ARNs específicos.

Por exemplo, para incluir `GetDevEndpoints` na política, o recurso deve estar projetado para o curinga (*). As operações singulares (`GetDevEndpoint`, `CreateDevEndpoint` e `DeleteDevendpoint`) também têm escopo para todos (*) os recursos no exemplo.

```
{
            "Sid": "PluralAPIIncluded",
            "Effect": "Allow",
            "Action": [
                "glue:GetDevEndpoints",
                "glue:GetDevEndpoint",
                "glue:CreateDevEndpoint",
                "glue:UpdateDevEndpoint"
            ],
            "Resource": [
                "*"
            ]
}
```

Controle de acesso para operações da API BatchGet fora do catálogo do AWS Glue

Algumas operações da API do AWS Glue recuperam vários itens (como vários endpoints de desenvolvimento); por exemplo, `BatchGetDevEndpoints`. Para essa operação, você pode especificar um ARN para limitar o escopo dos recursos que podem ser acessados.

Por exemplo, para permitir o acesso a um endpoint de desenvolvimento específico, inclua `BatchGetDevEndpoints` na política com o ARN do seu recurso.

```
{
            "Sid": "BatchGetAPIIncluded",
            "Effect": "Allow",
            "Action": [
                "glue:BatchGetDevEndpoints"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:123456789012:devEndpoint/de1"
            ]
}
```

Com essa política, você pode acessar com êxito o endpoint de desenvolvimento chamado `de1`. No entanto, se você tentar acessar o endpoint de desenvolvimento chamado `de2`, um erro será retornado.

```
An error occurred (AccessDeniedException) when calling the BatchGetDevEndpoints
 operation: No access to any requested resource.
```

> **Important**
>
> Para abordagens alternativas de configuração de políticas do IAM, como usar operações da API `List` e `BatchGet`, consulte Exemplos das políticas baseadas em identidade do AWS Glue.

Concessão de acesso entre contas

Conceder acesso aos recursos do Data Catalog entre contas permite que seus trabalhos de extração, transformação e carregamento (ETL) consultem e juntem dados de contas diferentes.

Tópicos

Métodos de concessão de acesso entre contas no AWS Glue

Você pode conceder acesso aos seus dados a contas externas da AWS usando métodos do AWS Glue ou concessões entre contas do AWS Lake Formation. Os métodos do AWS Glue usam políticas do AWS Identity and Access Management (IAM) para obter um controle de acesso minucioso. O Lake Formation usa um modelo de permissões `GRANT/REVOKE` mais simples semelhante aos comandos `GRANT/REVOKE` em um sistema de banco de dados relacional.

Esta seção descreve o uso de métodos do AWS Glue. Para mais informações sobre como usar concessões entre contas do Lake Formation, consulte Conceder permissões do Lake Formation no Guia do desenvolvedor do AWS Lake Formation.

Há dois métodos do AWS Glue para conceder acesso entre contas a um recurso:

- Usar uma política de recursos do Data Catalog
- Usar uma função do IAM

Usar uma política de recursos para conceder acesso entre contas

A seguir estão as etapas gerais para conceder acesso entre contas usando uma política de recursos do Data Catalog:

1. Um administrador (ou outra identidade autorizada) na conta A anexa uma política de recursos ao Data Catalog na conta A. Essa política concede à conta B permissões específicas entre contas para executar operações em um recurso no catálogo da conta A.
2. Um administrador na conta B anexa uma política do IAM a uma identidade do IAM na conta B que delega as permissões recebidas da conta A.

A identidade na conta B agora tem acesso ao recurso especificado na conta A.

O usuário precisa de permissão de ambos, o proprietário do recurso (conta A) e a conta superior (conta B), para poder acessar o recurso.

Conceder acesso entre contas utilizando uma função do IAM

A seguir estão as etapas gerais para conceder acesso entre contas usando uma função do IAM:

1. Um administrador (ou outra identidade autorizada) na conta que possui o recurso (conta A) cria uma função do IAM.
2. Um administrador na Conta A anexa uma política à função que concede permissões entre contas para acesso ao recurso em questão.
3. O administrador na conta A anexa uma política de confiança à função que identifica uma identidade do IAM em outra conta (conta B) como a entidade principal que pode assumir a função.

   A entidade principal da política de confiança também pode ser a entidade principal de um produto da AWS, se você desejar conceder permissão a um produto da AWS para que ele assuma a função.
4. Um administrador na conta B agora delega permissões para uma ou mais identidades do IAM na conta B, para que elas possam assumir essa função. Isso fornece às identidades na Conta B acesso ao recurso na conta A.

Para obter mais informações sobre o uso do IAM para delegar permissões, consulte Access management no IAM User Guide. Para obter mais informações sobre usuários, grupos, funções e permissões, consulte Identities (users, groups, and roles) no Guia do usuário do IAM.

Para obter uma comparação entre essas duas abordagens, consulte Como as funções do IAM diferem de políticas baseadas em recurso no Guia do usuário do IAM. O AWS Glue é compatível com ambas as opções, com a restrição de que uma política de recurso só pode conceder acesso aos recursos do Data Catalog.

Por exemplo, para conceder ao perfil `Dev` na conta B acesso ao banco de dados `db1` na conta A, anexe a política de recurso a seguir ao catálogo na conta 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"
      ]
    }
  ]
}
```

Além disso, a conta B deve anexar a seguinte política do IAM ao perfil `Dev`para poder realmente ter acesso ao db1 na conta 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"
      ]
    }
  ]
}
```

Adição ou atualização da política de recursos do catálogo de dados

Você pode adicionar ou atualizar a política de recursos do catálogo de dados do AWS Glue usando o console, a API ou a AWS Command Line Interface (AWS CLI).

> **Important**
>
> Se você já tiver feito concessões de permissão entre contas da sua conta com o AWS Lake Formation, adicionar ou atualizar a política de recursos do Data Catalog exigirá uma etapa extra. Para obter mais informações, consulte Gerenciar permissões entre contas usando o AWS Glue e o Lake Formation no Guia do desenvolvedor do AWS Lake Formation.
> Para determinar se existem concessões entre contas do Lake Formation, use a operação de API `glue:GetResourcePolicies` ou a AWS CLI. Se `glue:GetResourcePolicies` retornar qualquer outra política, exceto uma política já existente do Data Catalog, será porque existem concessões do Lake Formation. Para obter mais informações, consulte Visualizar todas as concessões entre contas usando a operação de API GetResourcePolicies no Guia do desenvolvedor do AWS Lake Formation.

Para adicionar ou atualizar a política de recursos do Data Catalog (console)

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.

   Faça login como um usuário administrativo do AWS Identity and Access Management (IAM) que tem a permissão de `glue:PutResourcePolicy`.
2. No painel de navegação, selecione Configurações.
3. Na página Data catalog settings (Configurações de catálogo de dados), em Permissions (Permissões), cole uma política de recursos na área de texto. Em seguida, escolha Salvar.

   Se o console exibir um alerta informando que as permissões na política serão adicionais às permissões concedidas usando o Lake Formation, escolha Proceed (Continuar).

Para adicionar ou atualizar a política de recursos do Data Catalog (AWS CLI)

- Envie um comando `aws glue put-resource-policy`. Se já existirem concessões do Lake Formation, certifique-se de incluir a opção `--enable-hybrid` com o valor `'TRUE'`.

  Para obter exemplos de uso dessa operação, consulte Exemplos de política baseada em recursos para o AWS Glue.

Realização de uma chamada de API entre contas

Todas as operações do AWS Glue Data Catalog têm um campo `CatalogId`. Se as permissões necessárias foram concedidas para habilitar o acesso entre contas, um autor da chamada pode fazer chamadas de API do Data Catalog entre contas. O autor da chamada faz isso transmitindo o ID da conta da AWS de destino no `CatalogId` para acessar o recurso nessa conta de destino.

Se nenhum valor de `CatalogId` for fornecido, o AWS Glue usará o ID da conta do próprio chamador por padrão, e a chamada não será entre contas.

Realização de uma chamada de ETL entre contas

Algumas APIs de PySpark e Scala do AWS Glue têm um campo de ID do catálogo. Se todas as permissões necessárias forem concedidas para habilitar o acesso entre contas, um trabalho de ETL poderá fazer chamadas de PySpark e Scala às operações da API entre contas transmitindo o ID da conta da AWS de destino no campo de ID do catálogo para acessar os recursos do Data Catalog em uma conta de destino.

Se nenhum valor de ID de catálogo for fornecido, o AWS Glue usará o ID da conta do próprio chamador por padrão, e a chamada não será entre contas.

Para APIs do PySpark que oferecem suporte ao `catalog_id`, consulte GlueContext classe. Para APIs do Scala que oferecem suporte ao `catalogId`, consulte AWS GlueAPIs Scala GlueContext .

O exemplo a seguir mostra as permissões exigidas pelo usuário autorizado para executar um trabalho de ETL. Neste exemplo, *`grantee-account-id`* é o `catalog-id` do cliente que executa a tarefa e *`grantor-account-id`* é o proprietário do recurso. Este exemplo concede permissão a todos os recursos do catálogo na conta do concessor. Para limitar o escopo de recursos concedidos, você pode fornecer ARNs específicos para o catálogo, o banco de dados, a tabela e a conexão.

```
{
    "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**
>
> Se uma tabela na conta do concessor apontar para um local do Amazon S3 que também está na conta do concessor, a função do IAM usada para executar um trabalho de ETL na conta do usuário autorizado deverá ter permissão para listar e obter objetos da conta do concessor.

Uma vez que o cliente na Conta A já tem permissão para criar e executar tarefas de ETL, as etapas básicas para configurar uma tarefa de ETL para acesso entre contas são:

1. Permita acesso a dados entre contas (ignore esta etapa se o acesso entre contas do Amazon S3 já estiver configurado).
   a. Atualize a política do bucket do Amazon S3 na conta B para permitir o acesso entre contas na conta A.
   b. Atualizar a política do IAM na Conta A para permitir o acesso ao bucket na Conta B.
2. Permita o acesso entre contas ao Data Catalog.
   a. Crie ou atualize a política de recursos anexada ao Data Catalog na conta B para permitir o acesso na conta A.
   b. Atualize a política do IAM na conta A para permitir o acesso ao Data Catalog na conta B.

Registro em log no CloudTrail entre contas

Quando um trabalho de extração, transformação e carregamento (ETL) do AWS Glue acessa os dados subjacentes de uma tabela do catálogo de dados compartilhada por concessões entre contas do AWS Lake Formation, há um comportamento de registro em log adicional do AWS CloudTrail.

Para efeitos desta discussão, a conta da AWS que compartilha a tabela é a conta de proprietário, já a conta com a qual a tabela é compartilhada é a conta de destinatário. Quando um trabalho de

ETL na conta de destinatário acessa dados da tabela na conta de proprietário, o evento de acesso a dados do CloudTrail que é adicionado aos logs da conta de destinatário é copiado para os logs do CloudTrail da conta de proprietário. Isso é para que as contas de proprietário possam rastrear acessos a dados pelas várias contas de destinatários. Por padrão, os eventos do CloudTrail não incluem um identificador de entidade principal legível (ARN da entidade principal). Um administrador na conta de destinatário pode optar por incluir o ARN da entidade principal nos logs.

Para obter mais informações, consulte Registro em log entre contas do CloudTrail no Guia do desenvolvedor do AWS Lake Formation.

Consulte também

- the section called “Registro e monitoramento”

Propriedade e faturamento de recursos entre contas

Quando um usuário em uma conta da AWS (conta A) cria um novo recurso, como um banco de dados em outra conta (conta B), esse recurso é propriedade da conta B, a conta em que ele foi criado. Um administrador na Conta B automaticamente recebe permissões completas para acessar o novo recurso, incluindo leitura, gravação e concessão de permissões de acesso a uma conta de terceiros. O usuário na Conta A pode acessar o recurso que acabou de criar somente se tiver as permissões apropriadas concedidas pela Conta B.

Os custos de armazenamento e outros custos que estão diretamente associados ao novo recurso são cobrados da Conta B, a proprietária do recurso. O custo das solicitações do usuário que criou o recurso é faturado na conta do solicitante, a Conta A.

Para obter mais informações sobre faturamento e preços do AWS Glue, consulte Como funciona a definição de preços da AWS.

Limitações de acesso entre contas

O acesso entre contas do AWS Glue tem as seguintes limitações:

- O acesso entre contas ao AWS Glue não é permitido se você criou bancos de dados e tabelas usando o Amazon Athena ou o Amazon Redshift Spectrum antes de uma região ter suporte ao AWS Glue e a conta do proprietário do recurso não ter migrado o catálogo de dados do

Amazon Athena para o AWS Glue. Você pode encontrar o status atual da migração usando o GetCatalogImportStatus (get_catalog_import_status). Para obter mais detalhes sobre como migrar um catálogo do Athena para o AWS Glue, consulte Processo detalhado do upgrade para o AWS Glue Data Catalog no Guia do usuário do Amazon Athena.

- O acesso entre contas é compatível somente para recursos do Data Catalog, incluindo bancos de dados, tabelas, funções definidas pelo usuário e conexões.
- O acesso entre contas ao catálogo de dados do Athena exige que você registre o catálogo como um recurso do `DataCatalog` do Athena. Para obter instruções, consulte Registrar um AWS Glue Data Catalog de outra conta no Guia do usuário do Amazon Athena.

Solução de problemas de identidade e acesso ao AWS Glue

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o AWS Glue e o IAM.

Tópicos

- Não estou autorizado a realizar uma ação no AWS Glue
- Não estou autorizado a realizar iam: PassRole
- Quero permitir que pessoas de fora da minha acessem meus Conta da AWS recursos do AWS Glue

Não estou autorizado a realizar uma ação no AWS Glue

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo *`my-example-widget`* fictício, mas não tem as permissões `glue:`*`GetWidget`* fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 glue:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso *`my-example-widget`* usando a ação `glue:`*`GetWidget`*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a realizar iam: PassRole

Se você receber um erro informando que não está autorizado a realizar a `iam:PassRole` ação, suas políticas devem ser atualizadas para permitir que você passe uma função para o AWS Glue.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para realizar uma ação no AWS Glue. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas de fora da minha acessem meus Conta da AWS recursos do AWS Glue

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o AWS Glue é compatível com esses recursos, consulteComo o AWS Glue funciona com o IAM.

- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre usar funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Registrar em log e monitorar no AWS Glue

Você pode automatizar a execução dos seus trabalhos de ETL (extração, transformação e carregamento). O AWS Glue fornece métricas para crawlers e trabalhos que você pode monitorar. Depois de configurar o AWS Glue Data Catalog com os metadados necessários, o AWS Glue fornecerá estatísticas sobre a integridade do seu ambiente. Você pode automatizar a invocação de crawlers e trabalhos com uma programação de cron baseada em hora. Você também pode ativar trabalhos quando um gatilho baseado em eventos é acionado.

O AWS Glue é integrado ao AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, por uma função ou por um produto da AWS no AWS Glue. Ao criar uma trilha, você poderá habilitar a entrega contínua de eventos do CloudTrail para um bucket do Amazon Simple Storage Service (Amazon S3), Amazon CloudWatch Logs e Amazon CloudWatch Events. Cada entrada de log ou evento contém informações sobre quem gerou a solicitação.

Use o Amazon CloudWatch Events para automatizar seus produtos da AWS e responder automaticamente a eventos do sistema, como problemas de disponibilidade de aplicação ou alterações de recursos. Os eventos dos produtos da AWS são entregues ao CloudWatch Events quase em tempo real. Você pode escrever regras simples para indicar quais eventos são interessantes e quais ações automatizadas devem ser tomadas quando um evento corresponder a uma regra.

Consulte também

- Automatizando o AWS Glue com o CloudWatch Events

- Registro em log no CloudTrail entre contas

Um aspecto importante da segurança na nuvem é o registro em log. Você deve configurar o registro em log de modo a não capturar segredos e material confidencial enquanto captura as informações necessárias para depurar e proteger sua estrutura de infraestrutura na nuvem. Não esqueça de se familiarizar com o que está sendo registrado em log.

Validação de conformidade para AWS Glue

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos AWS focados em segurança e conformidade.
- Arquitetura para segurança e conformidade com a HIPAA na Amazon Web Services — Este whitepaper descreve como as empresas podem usar AWS para criar aplicativos qualificados para a HIPAA.

> **Note**
>
> Nem todos Serviços da AWS são elegíveis para a HIPAA. Para obter mais informações, consulte a Referência dos serviços qualificados pela HIPAA.

- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- AWS Guias de conformidade do cliente — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da

AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).

- Avaliação de recursos com regras no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar os atributos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub.
- Amazon GuardDuty — Isso AWS service (Serviço da AWS) detecta possíveis ameaças às suas cargas de trabalho Contas da AWS, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, como o PCI DSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- AWS Audit Manager— Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Resiliência em AWS Glue

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que executam o failover automaticamente entre as zonas de disponibilidade sem interrupção. As Zonas de Disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de datacenter tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Para obter mais informações sobre resiliência do AWS Glue trabalho, consulte Erro: comportamento de failover entre VPCs em. AWS Glue

Segurança da infraestrutura no AWS Glue

Como um serviço gerenciado, o AWS Glue é protegido pelos procedimentos de segurança de rede global da AWS que estão descritos no whitepaper Amazon Web Services: Overview of Security Processes.

Você usa AWS Glue chamadas de API publicadas pela para acessarAWS o por meio da rede. Os clientes devem oferecer suporte a Transport Layer Security (TLS) 1.0 ou posterior. Recomendamos TLS 1.2 ou posterior. Os clientes também devem ter suporte a conjuntos de criptografia com perfect forward secrecy (PFS) como Ephemeral Diffie-Hellman (DHE) ou Ephemeral Elliptic Curve Diffie-Hellman (ECDHE). A maioria dos sistemas modernos como Java 7 e versões posteriores oferece suporte a esses modos.

Além disso, as solicitações devem ser assinadas utilizando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Tópicos

- AWS Glue e endpoint da VPC de interface (AWS PrivateLink)
- Amazon VPCs compartilhadas

AWS Glue e endpoint da VPC de interface (AWS PrivateLink)

É possível estabelecer uma conexão privada entre a VPC e o AWS Glue criando um endpoint da VPC de interface. Os endpoints de interface são habilitados por AWS PrivateLink, uma tecnologia que permite acessar de forma privada as APIs do AWS Glue sem um gateway da Internet, um dispositivo NAT, uma conexão VPN ou uma conexão do AWS Direct Connect. As instâncias na VPC não precisam de endereços IP públicos para a comunicação com APIs do AWS Glue. O tráfego de rede entre a VPC e o AWS Glue não deixa a rede da Amazon.

Cada endpoint de interface é representado por uma ou mais interfaces de rede elástica nas sub-redes.

Para mais informações, consulte VPC endpoints de interface (AWS PrivateLink) no Guia do usuário da Amazon VPC.

Considerações sobre endpoints da VPC do AWS Glue

Antes de configurar um endpoint da VPC de interface para AWS Glue, certifique-se de revisar as Propriedades e limitações do endpoint de interface no Guia do Usuário da Amazon VPC.

O AWS Glue oferece suporte a chamadas para todas as ações de API da VPC.

Criar um endpoint da VPC de interface para o AWS Glue

É possível criar um endpoint da VPC para o serviço AWS Glue usando o console do Amazon VPC ou a AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte Criar um endpoint de interface no Guia do Usuário do Amazon VPC.

Crie um endpoint da VPC para o AWS Glue usando o seguinte nome de serviço:

- com.amazonaws.*região*.glue

Se você habilitar o DNS privado para o endpoint, poderá fazer solicitações de API para o AWS Glue usando seu nome DNS padrão para a região, por exemplo, `glue.us-east-1.amazonaws.com`.

Para obter mais informações, consulte Acessar um serviço por um endpoint de interface no Manual do Usuário do Amazon VPC.

Criando uma política de endpoint da VPC para o AWS Glue

É possível anexar uma política de endpoint ao endpoint da VPC que controla o acesso ao AWS Glue. Essa política especifica as seguintes informações:

- A entidade principal que pode executar ações.
- As ações que podem ser executadas.
- Os recursos sobre os quais as ações podem ser realizadas.

Para mais informações, consulte Controlar o acesso a serviços com VPC endpoints no Guia do usuário da Amazon VPC.

Exemplo: política de endpoint de VPC para o AWS Glue para permitir a criação e atualização do trabalho

Veja a seguir um exemplo de uma política de endpoint para o AWS Glue. Quando anexada a um endpoint, essa política concede acesso às ações indicadas do AWS Glue para todos os principais em todos os recursos.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
             "glue:CreateJob",
             "glue:UpdateJob",
             "iam:PassRole"
         ],
         "Resource":"*"
      }
   ]
}
```

Exemplo: política de endpoint de VPC para permitir acesso somente de leitura ao catálogo de dados

Veja a seguir um exemplo de uma política de endpoint para o AWS Glue. Quando anexada a um endpoint, essa política concede acesso às ações indicadas do AWS Glue para todos os principais em todos os 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": "*"
        }
    ]
}
```

Amazon VPCs compartilhadas

O AWS Glue é compatível com nuvens privadas virtuais (VPCs) compartilhadas na Amazon Virtual Private Cloud. O compartilhamento da Amazon VPC permite que várias contas da AWS criem os recursos de aplicações, como instâncias do Amazon EC2 e bancos de dados do Amazon Relational Database Service (Amazon RDS), em Amazon VPCs compartilhadas e gerenciadas centralmente. Nesse modelo, a conta que possui a VPC (proprietária) compartilha uma ou mais sub-redes com outras contas (participantes) que pertencem à mesma organização no AWS Organizations. Quando uma sub-rede é compartilhada, os participantes podem visualizar, criar, modificar e excluir os recursos de aplicativos nas sub-redes compartilhadas com eles.

No AWS Glue, para criar uma conexão com uma sub-rede compartilhada, é necessário criar um grupo de segurança em sua conta e anexá-lo à sub-rede compartilhada.

Para obter mais informações, consulte estes tópicos:

- Trabalhar com VPCs compartilhadas no Manual do usuário da Amazon VPC
- O que é o AWS Organizations? no Manual do usuário do AWS Organizations

Solução de problemas de AWS Glue

Tópicos

Reunir informações sobre a solução de problemas do AWS Glue

Se você encontrar erros ou um comportamento inesperado no AWS Glue e precisar entrar em contato com o AWS Support, reúnas as informações sobre nomes, IDs e logs associados à ação com falha. A disponibilização dessas informações permite que o AWS Support ajude você a solucionar os problemas.

Além do seu ID de conta, forneça as seguintes informações para cada um desses tipos de falhas:

Quando um crawler falhar, reúna as seguintes informações:

- Nome do crawler

  Os logs das execuções de crawlers estão localizados no CloudWatch Logs, em `/aws-glue/crawlers`.

Quando uma conexão de teste falhar, reúna as seguintes informações:

- Connection name (Nome da conexão)
- ID da conexão
- String de conexão do JDBC no formato `jdbc:protocol://host:port/database-name`.

  Os logs das conexões de teste estão localizados no CloudWatch Logs, em `/aws-glue/testconnection`.

Quando um trabalho falhar, reúna as seguintes informações:

- Nome do trabalho
- ID de execução do trabalho, no formato `jr_xxxxx`.

Os logs das execuções de trabalhos estão localizados no CloudWatch Logs, em `/aws-glue/jobs`.

Solução de problemas de erros no AWS Glue Spark

Se você encontrar erros no AWS Glue, use as soluções a seguir para ajudá-lo a encontrar a origem dos problemas e corrigi-los.

> **Note**
>
> O AWS Glue GitHub repositório contém orientações adicionais para solução de problemas em AWS Glue Perguntas frequentes.

Tópicos

Erro: Resource unavailable (Recurso indisponível)

Se AWS Glue retornar uma mensagem de recurso indisponível, você poderá visualizar mensagens de erro ou registros para ajudá-lo a saber mais sobre o problema. As tarefas a seguir descrevem métodos gerais para a solução de um problema.

- Para quaisquer conexões e endpoints de desenvolvimento que você usar, verifique se o seu cluster não ficou sem interfaces de rede elástica.

Erro: Could not find S3 endpoint or NAT gateway for subnetId in VPC (Não foi possível encontrar o endpoint do S3 nem o gateway NAT para subnetId na VPC)

Verifique o ID da sub-rede e o ID da VPC na mensagem para diagnosticar o problema.

- Verifique se você possui um endpoint da VPC do Amazon S3 configurado (necessário para uso com o AWS Glue). Além disso, verifique seu gateway NAT se ele for parte da configuração. Para ter mais informações, consulte Endpoints da Amazon VPC para o Amazon S3.

Erro: Inbound rule in security group required (A regra de entrada no grupo de segurança é obrigatória)

Pelo menos um security group precisa abrir todas as portas de entrada. Para limitar o tráfego, o security group de origem na sua regra de entrada pode ser restrito ao mesmo security group.

- Para todas as conexões que você usar, verifique se seu security group contém uma regra de entrada de autorreferenciada. Para ter mais informações, consulte Configurar o acesso de rede aos armazenamentos de dados.
- Ao usar um endpoint de desenvolvimento, verifique se seu security group contém para uma regra de entrada de autorreferenciada. Para ter mais informações, consulte Configurar o acesso de rede aos armazenamentos de dados.

Erro: Outbound rule in security group required (A regra de saída no grupo de segurança é obrigatória)

Pelo menos um security group precisa abrir todas as portas de saída. Para limitar o tráfego, o security group de origem na sua regra de saída pode ser restrito ao mesmo security group.

- Para todas as conexões que você usar, verifique se seu security group contém uma regra de saída de autorreferenciada. Para ter mais informações, consulte Configurar o acesso de rede aos armazenamentos de dados.
- Ao usar um endpoint de desenvolvimento, verifique se seu security group contém para uma regra de saída de autorreferenciada. Para ter mais informações, consulte Configurar o acesso de rede aos armazenamentos de dados.

Erro: Falha na execução do trabalho porque a função passada deve receber permissões de assumir função para o AWS Glue serviço

O usuário que define um trabalho precisa ter permissão para `iam:PassRole` no AWS Glue.

- Quando um usuário cria um AWS Glue trabalho, confirme se a função do usuário contém uma política que contém `iam:PassRole` for AWS Glue. Para ter mais informações, consulte Etapa 3: anexar uma política aos usuários ou grupos que acessam o AWS Glue.

Erro: a DescribeVpcEndpoints ação não foi autorizada. não foi possível validar a ID da VPC vpc-id

- Verifique a política aprovada AWS Glue para obter a `ec2:DescribeVpcEndpoints` permissão.

Erro: a DescribeRouteTables ação não foi autorizada. não foi possível validar a ID da sub-rede: ID da sub-rede na VPC ID: vpc-id

- Verifique a política aprovada AWS Glue para obter a `ec2:DescribeRouteTables` permissão.

Erro: falha ao chamar ec2: DescribeSubnets

- Verifique a política aprovada AWS Glue para obter a `ec2:DescribeSubnets` permissão.

Erro: falha ao chamar ec2: DescribeSecurityGroups

- Verifique a política aprovada AWS Glue para obter a `ec2:DescribeSecurityGroups` permissão.

Erro: Could not find subnet for AZ (Não foi possível encontrar uma sub-rede para zona de disponibilidade)

- A zona de disponibilidade pode não estar disponível para AWS Glue. Crie e use uma nova sub-rede em uma zona de disponibilidade diferente da especificada na mensagem.

Erro: Job run exception when writing to a JDBC target (Exceção de execução de trabalho ao gravar um destino JDBC)

Ao executar um trabalho que grava em um destino JDBC, ele pode encontrar erros nas seguintes situações:

- Se o seu trabalho gravar em uma tabela do Microsoft SQL Server, e a tabela possuir colunas definidas como tipo `Boolean`, ela deverá ser predefinida no banco de dados do SQL Server. Ao definir o trabalho no AWS Glue console usando um destino do SQL Server com a opção Criar tabelas em seu destino de dados, não mapeie nenhuma coluna de origem para uma coluna de destino com tipo de dados`Boolean`. Você poderá encontrar um erro quando o trabalho for executado.

  Você pode evitar fazendo o seguinte:

  - Escolha uma tabela existente com a coluna Boolean.
  - Edite a transformação `ApplyMapping` e mapeie a coluna Boolean na fonte para um número ou uma string no destino.
  - Edite a transformação `ApplyMapping` para remover a coluna Boolean da fonte.
- Se o seu trabalho escrever em uma tabela Oracle, será necessário ajustar o comprimento dos nomes dos objetos Oracle. Em algumas versões do Oracle, o comprimento máximo do identificador é limitado a 30 ou 128 bytes. Esse limite afeta os nomes das tabelas e os nomes das colunas dos armazenamentos de dados de destino Oracle.

  Você pode evitar fazendo o seguinte:

  - Nomeie as tabelas de destino Oracle de acordo com o limite da sua versão.
  - Os nomes de coluna padrão são gerados a partir dos nomes dos campos nos dados. Quando os nomes das colunas forem mais longos do que o limite, use transformações `ApplyMapping` ou `RenameField` para alterar o nome da coluna de modo que ela fique dentro do limite.

Erro: Amazon S3: a operação não é válida para a classe de armazenamento do objeto

Se AWS Glue retornar esse erro, seu AWS Glue trabalho pode estar lendo dados de tabelas que têm partições em vários níveis de classe de armazenamento do Amazon S3.

- Ao usar exclusões de classe de armazenamento, você pode garantir que seus AWS Glue trabalhos funcionem em tabelas que tenham partições nesses níveis de classe de armazenamento. Sem exclusões, trabalhos que leem dados desses níveis falham com o seguinte erro: `AmazonS3Exception: The operation is not valid for the object's storage class`.

  Para ter mais informações, consulte Excluir classes de armazenamento do Amazon S3.

Erro: Amazon S3 timeout (Tempo limite do Amazon S3)

Se AWS Glue retornar um erro de tempo limite de conexão, pode ser porque ele está tentando acessar um bucket do Amazon S3 em AWS outra região.

- Um endpoint VPC do Amazon S3 só pode rotear tráfego para buckets dentro de uma região. AWS Se você precisar se conectar aos buckets em outras regiões, tente usar um gateway NAT como uma solução alternativa. Para obter mais informações, consulte Gateways NAT.

Erro: Amazon S3 access denied (Acesso negado ao Amazon S3)

Se AWS Glue retornar um erro de acesso negado a um bucket ou objeto do Amazon S3, pode ser porque a função do IAM fornecida não tem uma política com permissão para seu armazenamento de dados.

- Um trabalho de ETL precisa de acesso ao datastore do Amazon S3 usado como fonte ou destino. Um crawler precisa de acesso ao datastore do Amazon S3 que ele rastreia. Para ter mais informações, consulte Etapa 2: criar um perfil do IAM para o AWS Glue.

Erro: Amazon S3 access key ID does not exist (O ID da chave de acesso do Amazon S3 não existe)

Se AWS Glue retornar um erro de ID de chave de acesso não existe ao executar um trabalho, pode ser devido a um dos seguintes motivos:

- Um trabalho de ETL usa uma função do IAM para acessar os armazenamentos de dados. Verifique se a função do IAM para o seu trabalho não foi excluída antes do início dele.
- Uma função do IAM contém permissões para acessar seus armazenamentos de dados. Verifique se as políticas anexadas do Amazon S3 que contêm `s3:ListBucket` estão corretas.

Erro: falha na execução do trabalho ao acessar o Amazon S3 com um URI **s3a://**

Se a execução de um trabalho retornar um erro como Failed to parse XML document with handler class (Falha ao analisar um documento XML com a classe handler), pode ser devido a uma falha ao tentar listar centenas de arquivos usando um URI `s3a://`. Acesse seu datastore usando um URI `s3://`. O rastreamento de exceções a seguir destaca os erros a serem procurados:

```
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)
```

Erro: Amazon S3 service token expired (O token do serviço do Amazon S3 expirou)

Ao mover dados de e para o Amazon Redshift, as credenciais temporárias do Amazon S3, que expiram após uma hora, são usadas. Se você tiver um trabalho de longa execução, ele poderá falhar. Para obter informações sobre como configurar seus trabalhos de longa duração de execução para mover dados de e para o Amazon Redshift, consulte aws-glue-programming-etl-connect-redshift-home.

Erro: No private DNS for network interface found (Nenhum DNS privado foi encontrado na interface de rede)

Se um trabalho falhar ou um endpoint de desenvolvimento não for provisionado, é possível que tenha ocorrido um problema na configuração da rede.

- Se você estiver usando o DNS fornecido pela Amazon, o valor de `enableDnsHostnames` deverá ser true. Para mais informações, consulte DNS.

Erro: Development endpoint provisioning failed (Falha no provisionamento do endpoint de desenvolvimento)

Se AWS Glue não conseguir provisionar com êxito um endpoint de desenvolvimento, pode ser devido a um problema na configuração da rede.

- Ao definir um endpoint de desenvolvimento, a VPC, a sub-rede e os security groups são validados para confirmar se atendem a determinados requisitos.
- Se você forneceu a chave pública SSH opcional, verifique se ela está válida.
- Verifique no console da VPC se a sua VPC está usando um DHCP option set válido. Para obter mais informações, consulte Conjuntos de opções de DHCP.
- Se o cluster permanecer no estado PROVISIONING (Provisionamento), entre em contato com o AWS Support.

Erro: Notebook server CREATE_FAILED (Servidor de cadernos com status CREATE_FAILED)

Se AWS Glue não conseguir criar o servidor do notebook para um endpoint de desenvolvimento, pode ser devido a um dos seguintes problemas:

- AWS Glue passa uma função do IAM para o Amazon EC2 quando ele está configurando o servidor do notebook. A função do IAM precisa ter uma relação de confiança com o Amazon EC2.
- A função do IAM precisa ter um perfil da instância com o mesmo nome. Quando você cria a função para o Amazon EC2 usando o console do IAM, o perfil da instância com o mesmo nome é automaticamente criado. Verifique se há um erro no log referente a um nome de perfil da instância `iamInstanceProfile.name` que seja inválido. Para obter mais informações, consulte Usar perfis de instância.
- Verifique se sua função tem permissão para acessar buckets do `aws-glue*` na política que você transmitiu para criar o servidor da notebook.

Erro: Local notebook fails to start (Falha ao iniciar o caderno local)

Se o seu notebook local não for iniciado houver indicação de erros referentes a diretórios ou pastas não encontrados, é possível que um dos seguintes problemas tenha ocorrido:

- Se você estiver utilizando o Microsoft Windows, verifique se a variável de ambiente `JAVA_HOME` aponta para o diretório correto do Java. É possível atualizar o Java sem atualizar essa variável e, se ela apontar para uma pasta que não existe mais, os cadernos Jupyter não serão inicializados.

Erro: Running crawler failed (Falha na execução do crawler)

Se AWS Glue não conseguir executar com êxito um rastreador para catalogar seus dados, pode ser devido a um dos seguintes motivos. Verifique primeiro se o erro está presente na lista de crawlers do console do AWS Glue . Verifique se há um ícone de exclamação ao lado do nome do crawler e posicione o mouse sobre o ícone para ver as mensagens associadas.

- Verifique os registros do rastreador executado em CloudWatch Registros, em. `/aws-glue/crawlers`

Erro: Partitions were not updated (As partições não foram atualizadas)

Caso suas partições não tenham sido atualizadas no Catálogo de Dados quando você executou um trabalho de ETL, essas instruções de log da `DataSink` classe nos CloudWatch registros podem ser úteis:

- `"Attempting to fast-forward updates to the Catalog - nameSpace:"` – mostra qual banco de dados, tabela e catalogId este trabalho tentará modificar. Se essa instrução não estiver aqui, verifique se `enableUpdateCatalog` está definido como verdadeiro e corretamente passado como um parâmetro `getSink()` ou em `additional_options`.
- `"Schema change policy behavior:"` – mostra qual valor `updateBehavior` do esquema foi passado.
- `"Schemas qualify (schema compare):"` – será verdadeiro ou falso.
- `"Schemas qualify (case-insensitive compare):"` – será verdadeiro ou falso.
- Se ambos forem falsos e o seu não `updateBehavior` estiver definido como `UPDATE_IN_DATABASE`, seu DynamicFrame esquema precisará ser idêntico ou conter um subconjunto das colunas vistas no esquema da tabela do Catálogo de Dados.

Para obter mais informações sobre a atualização de partições, consulte Atualizar esquemas e adicionar novas partições ao Catálogo de Dados em trabalhos do AWS Glue ETL.

Erro: Job bookmark update failed due to version mismatch (A atualização do marcador de trabalho falhou devido à incompatibilidade da versão)

Talvez você esteja tentando parametrizar AWS Glue trabalhos para aplicar a mesma transformação/lógica em diferentes conjuntos de dados no Amazon S3. Você deseja rastrear arquivos processados nos locais fornecidos. Quando você executa o mesmo trabalho no mesmo bucket de origem e grava simultaneamente no mesmo/em outro destino (simultaneidade >1), o trabalho falha com este erro:

```
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
```

Solução: defina a simultaneidade como 1 ou não execute o trabalho simultaneamente.

Atualmente, os AWS Glue favoritos não suportam a execução simultânea de trabalhos e os commits falharão.

Erro: A job is reprocessing data when job bookmarks are enabled (Um trabalho está reprocessando dados quando marcadores do trabalho estão habilitados)

Pode haver casos em que você tenha habilitado os marcadores de AWS Glue tarefas, mas sua tarefa de ETL esteja reprocessando dados que já foram processados em uma execução anterior. Verifique se essas causas comuns desse erro existem:

Simultaneidade máxima

Definir o número máximo de execuções simultâneas para o trabalho maior que o valor padrão de 1 pode interferir nos marcadores do trabalho. Isso pode ocorrer quando marcadores de trabalhos verificam a hora da última modificação de objetos para verificar quais objetos precisam ser reprocessados. Para obter mais informações, consulte a discussão sobre simultaneidade máxima em Configurando as propriedades da tarefa para tarefas do Spark no AWS Glue.

Objeto de trabalho ausente

Verifique se o script de execução do trabalho termina com a seguinte confirmação:

```
job.commit()
```

Quando você inclui esse objeto, AWS Glue registra a data e hora e o caminho da execução do trabalho. Se você executar o trabalho novamente com o mesmo caminho, AWS Glue processará somente os novos arquivos. Se você não incluir esse objeto, e os marcadores do trabalho estiverem habilitados, o trabalho reprocessará os arquivos já processados junto com os arquivos novos e criará redundância no datastore de destino do trabalho.

Parâmetro de contexto da transformação ausente

O contexto de transformação é um parâmetro opcional na classe `GlueContext`, mas os marcadores de trabalho não funcionarão se você não o incluir. Para resolver esse erro, adicione o parâmetro de contexto de transformação ao criar o DynamicFrame, conforme mostrado a seguir:

```
sample_dynF=create_dynamic_frame_from_catalog(database,
 table_name,transformation_ctx="sample_dynF")
```

Fonte de entrada

Se você estiver usando um banco de dados relacional (uma conexão JDBC) para a fonte de entrada, os marcadores de trabalho só funcionarão se as chaves primárias da tabela estiverem em ordem sequencial. Os marcadores de trabalhos funcionam para linhas novas, mas não para linhas atualizadas. Isso ocorre porque os marcadores de trabalho procuram as chaves primárias, que já existem. Isso não se aplica se a fonte de entrada for o Amazon Simple Storage Service (Amazon S3).

Hora da última modificação

Para fontes de entrada do Amazon S3, os marcadores de trabalho verificam a hora da última modificação dos objetos, em vez dos nomes de arquivo, para verificar quais objetos precisam ser reprocessados. Se os dados da fonte de entrada tiverem sido modificados desde a última execução do trabalho, os arquivos serão reprocessados quando você executar o trabalho novamente.

Erro: comportamento de failover entre VPCs em AWS Glue

O processo a seguir é usado para failover para trabalhos na AWS Glue versão 4.0 e nas versões anteriores.

Resumo: uma AWS Glue conexão é selecionada no momento em que uma execução de trabalho é enviada. Se a execução do trabalho encontrar alguns problemas (falta de endereços IP, conectividade com a fonte, problema de roteamento), a execução do trabalho falhará. Se as novas tentativas estiverem configuradas, AWS Glue tentará novamente com a mesma conexão.

1. Para cada tentativa de execução, AWS Glue verificará a integridade das conexões na ordem listada na configuração do trabalho, até encontrar uma que possa ser usada. No caso de uma falha na Zona de Disponibilidade (AZ), as conexões dessa AZ falharão na verificação e serão ignoradas.
2. AWS Glue valida a conexão com o seguinte:
   - verifica se existem ID e sub-rede da Amazon VPC válidos.
   - verifica se existe um gateway NAT ou um endpoint da Amazon VPC.
   - verifica se a sub-rede tem mais de 0 endereços IP alocados.
   - verifica se a AZ está íntegra.

   AWS Glue não é possível verificar a conectividade no momento do envio da execução do trabalho.

3. Para trabalhos usando a Amazon VPC, todos os drivers e executores serão criados na mesma AZ com a conexão selecionada no momento do envio da execução do trabalho.
4. Se as novas tentativas estiverem configuradas, AWS Glue tentará novamente com a mesma conexão. Isso ocorre porque não podemos garantir que os problemas com essa conexão sejam de longa duração. Se uma AZ falhar, as execuções de trabalhos existentes (dependendo do estágio da execução do trabalho) nessa AZ poderá falhar. Uma nova tentativa deve detectar uma falha de AZ e escolher outra AZ para a nova execução.

Solucionar problemas de erros do crawler quando o crawler estiver usando credenciais do Lake Formation

Use as informações abaixo para diagnosticar e corrigir vários problemas ao configurar o crawler usando as credenciais do Lake Formation.

Erro: The S3 location: s3://examplepath is not registered (O local do S3: s3://examplepath não está registrado)

Para que um crawler seja executado usando as credenciais do Lake Formation, você precisa primeiro configurar as permissões do Lake Formation. Para resolver esse erro, registre o local de destino do Amazon S3 no Lake Formation. Para obter mais informações, consulte Registering an Amazon S3 location (Registrar um local do Amazon S3).

Erro: User/Role is not authorized to perform: lakeformation:GetDataAccess on resource (O usuário/função não está autorizado(a) a executar lakeformation:GetDataAccess no recurso)

Adicione a permissão `lakeformation:GetDataAccess` à função do crawler usando o console do IAM ou a AWS CLI. Com essa permissão, o Lake Formation concede a solicitação de credenciais temporárias para acessar os dados. Consulte a política abaixo:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "lakeformation:GetDataAccess"
    ],
```

```
    "Resource": "*"
  }
}
```

Erro: Insufficient Lake Formation permission(s) on (Database name: exampleDatabase, Table Name: exampleTable) (Permissão(ões) do Lake Formation insuficiente(s) em (Nome do banco de dados: exampleDatabase, Nome da tabela: exampleTable))

No console do Lake Formation (https://console.aws.amazon.com/lakeformation/), conceda permissões de acesso à função do crawler (`Create`, `Describe`, `Alter`) no banco de dados, que é especificado como o banco de dados de saída. Você também pode conceder permissões na tabela. Para obter mais informações, consulte Granting database permissions using the named resource method (Conceder permissões de banco de dados usando o método de recurso nomeado).

Erro: Insufficient Lake Formation permission(s) on s3://examplepath (Permissão(ões) insuficiente(s) do Lake Formation em s3://examplepath)

1. Crawling entre contas
   a. Faça login no console do Lake Formation (https://console.aws.amazon.com/lakeformation/) usando a conta em que o bucket do Amazon S3 está registrado (conta B). Conceda permissões de localização de dados para a conta em que o crawler será executado. Isso permitirá que o crawler leia os dados da localização de destino do Amazon S3.
   b. Na conta na qual o crawler é criado (conta A), conceda permissões de localização de dados na localização de destino do Amazon S3 ao perfil do IAM usado para a execução do crawler, de modo que o crawler possa ler os dados do destino no Lake Formation. Para obter mais informações, consulte Granting data location permissions (same account) (Conceder permissões de localização de dados (mesma conta)).
2. Crawling na conta (crawler e localização registrada do Amazon S3 estão na mesma conta): conceda permissões de localização de dados ao perfil do IAM usado para a execução do crawler, de modo que o crawler possa ler os dados do destino no Lake Formation. Para obter mais informações, consulte Granting data location permissions (same account) (Conceder permissões de localização de dados [mesma conta]).

Perguntas frequentes sobre a configuração do crawler usando as credenciais do Lake Formation

1. Como faço para configurar um crawler para ser executado usando as credenciais do Lake Formation por meio do console da AWS?

   No console do AWS Glue (https://console.aws.amazon.com/glue/), ao configurar o crawler, selecione a opção Use Lake Formation credentials for crawling Amazon S3 data source (Usar credenciais do Lake Formation para crawling da fonte de dados do Amazon S3). Para crawling entre contas, especifique o ID da Conta da AWS na qual a localização de destino do Amazon S3 está registrada no Lake Formation. O campo accountId é opcional para crawling na conta.

2. Como faço para configurar um crawler para ser executado usando as credenciais do Lake Formation por meio da AWS CLI?

   Durante a chamada da API `CreateCrawler`, adicione `LakeFormationConfiguration`:

   ```
   "LakeFormationConfiguration": {
       "UseLakeFormationCredentials": true,
       "AccountId": "111111111111" (AWS account ID where the target Amazon S3 location
    is registered with Lake Formation)
     }
   ```

3. Quais são os destinos compatíveis com um crawler que usa as credenciais do Lake Formation?

   Um crawler que usa credenciais do Lake Formation só é compatível com destinos do Amazon S3 (crawling na conta e entre contas) e do catálogo de dados na conta (onde a localização subjacente é o Amazon S3) e em destinos do Apache Iceberg.

4. Posso fazer crawling de vários buckets do Amazon S3 como parte de um único crawler usando as credenciais do Lake Formation?

   Não. Para destinos de crawling que utilizam o fornecimento de credenciais do Lake Formation, as localizações subjacentes do Amazon S3 devem pertencer ao mesmo bucket. Por exemplo, os clientes podem usar vários locais de destino (`s3://bucket1/folder1, s3://bucket1/folder2`) se estiverem no mesmo bucket (bucket1). Não existe suporte para a especificação de buckets diferentes (s3://bucket1/folder1, s3://bucket2/folder2).

Solucionar problemas do AWS Glue para erros do Ray a partir de logs

O AWS Glue dá acesso aos logs emitidos pelos processos do Ray durante a execução do trabalho. Se você encontrar erros ou comportamentos inesperados nos trabalhos do Ray, primeiro colete as informações dos logs para determinar a causa da falha. Também fornecemos logs semelhantes para sessões interativas. Os logs das sessões são fornecidos com o prefixo `/aws-glue/ray/sessions`.

As linhas de log são enviadas para o CloudWatch em tempo real, à medida que seu trabalho é executado. As instruções de impressão são anexadas aos registros do CloudWatch após a conclusão da execução. Os registros são retidos por duas semanas após a execução de um trabalho.

Inspecionar logs de tarefas do Ray

Quando um trabalho falhar, colete o nome e o ID de execução do trabalho. É possível encontrá-los no console do AWS Glue. Navegue até a página do trabalho e, em seguida, navegue até a guia Runs (Execuções). Os logs de tarefas do Ray são armazenados nos seguintes grupos de logs dedicados do CloudWatch.

- `/aws-glue/ray/jobs/script-log/`: armazena logs emitidos pelo seu principal script do Ray.
- `/aws-glue/ray/jobs/ray-monitor-log/`: armazena logs emitidos pelo processo do autoscaler do Ray. Esses logs são gerados para o nó principal e não para outros nós de processamento.
- `/aws-glue/ray/jobs/ray-gcs-logs/`: armazena logs emitidos pelo processo GCS (loja de controle global). Esses logs são gerados para o nó principal e não para outros nós de processamento.
- `/aws-glue/ray/jobs/ray-process-logs/`: armazena logs emitidos por outros processos do Ray (principalmente o agente do painel) em execução no nó principal. Esses logs são gerados para o nó principal e não para outros nós de processamento.
- `/aws-glue/ray/jobs/ray-raylet-logs/`: Armazena logs emitidos por cada processo raylet. Esses logs são coletados em um único fluxo para cada nó de processamento, incluindo o nó principal.

- `/aws-glue/ray/jobs/ray-worker-out-logs/`: armazena logs do `stdout` para cada operador no cluster. Esses logs são gerados para cada nó de processamento, incluindo o nó principal.
- `/aws-glue/ray/jobs/ray-worker-err-logs/`: armazena logs do `stderr` para cada operador no cluster. Esses logs são gerados para cada nó de processamento, incluindo o nó principal.
- `/aws-glue/ray/jobs/ray-runtime-env-log/`: armazena logs sobre o processo de configuração do Ray. Esses logs são gerados para cada nó de processamento, incluindo o nó principal.

Solucionar de problemas de erros do Ray

Para entender a organização dos grupos de logs do Ray e encontrar os grupos de logs que ajudarão você a solucionar os erros, é útil ter informações básicas sobre a arquitetura do Ray.

No ETL AWS Glue, um proceesador corresponde a uma instância. Ao configurar processadores para um trabalho do AWS Glue, você está definindo o tipo e a quantidade de instâncias dedicadas ao trabalho. O Ray usa o termo processador de maneiras diferentes.

O Ray usa o nó principal e o nó de processamento para distinguir as responsabilidades de uma instância em um cluster do Ray. Um nó de processamento do Ray pode hospedar vários processos de atores que realizam cálculos para obter o resultado da computação distribuída. Os atores que executam uma réplica de uma função são chamados de réplicas. Os atores de réplica também podem ser chamados de processos de processador. As réplicas também podem ser executadas no nó principal, conhecido como principal porque executa processos adicionais para coordenar o cluster.

Cada ator que contribui para a computação gera seu próprio fluxo de logs. Isso nos fornece alguns insights:

- O número de processos que emitem logs pode ser maior que o número de processadores que podem ser alocados ao trabalho. Muitas vezes, cada núcleo em cada instância tem um ator.
- Os nós principais do Ray emitem logs de gerenciamento e de inicialização de cluster. Por outro lado, os nós de processamento do Ray só emitem logs para o trabalho realizado neles.

Para obter mais informações sobre a arquitetura Ray, consulte Architecture Whitepapers (Documentos técnicos de arquitetura) na documentação do Ray.

Área problemática: acesso ao Amazon S3

Verifique a mensagem de falha da execução de trabalho. Se isso não fornecer informações suficientes, verifique `/aws-glue/ray/jobs/script-log/`.

Área problemática: gerenciamento de dependências de PIP

Verifique `/aws-glue/ray/jobs/ray-runtime-env-log/`.

Área problemática: inspecionar valores intermediários no processo principal

Escreva para `stderr` ou `stdout` a partir do seu script principal e recupere logs de `/aws-glue/ray/jobs/script-log/`.

Área problemática: inspecionar valores intermediários no processo secundário

Escreva para `stderr` ou `stdout` partir da função `remote`. Em seguida, recupere os logs de `/aws-glue/ray/jobs/ray-worker-out-logs/` ou `/aws-glue/ray/jobs/ray-worker-err-logs/`. A função pode ter sido executada em qualquer réplica, então pode ser necessário examinar vários logs para encontrar a saída pretendida.

Área de problema: interpretação de endereços IP em mensagens de erro

Em determinadas situações de erro, seu trabalho pode emitir uma mensagem de erro contendo um endereço IP. Esses endereços IP são efêmeros e são usados pelo cluster para identificação e comunicação entre os nós. Os logs de um nó serão publicados em um fluxo de logs com um sufixo exclusivo baseado no endereço IP.

No CloudWatch, você pode filtrar os logs para inspecionar aqueles específicos desse endereço IP identificando esse sufixo. Por exemplo, considerando *`FAILED_IP`* e *`JOB_RUN_ID`*, você pode identificar o sufixo com:

```
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
```

Exceções de machine learning do AWS Glue

Este tópico descreve strings e códigos de erro HTTP para exceções do AWS Glue relacionadas a machine learning. Os códigos de erro e as strings de erro são fornecidos para cada atividade de machine learning que pode ocorrer quando você executa uma operação. Além disso, você pode ver se é possível repetir a operação que resultou no erro.

CancelMLTaskRunActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."
  - "Nenhuma execução de tarefa de ML encontrada para [taskRunId]: na conta [accountId] para a transformação [transformName]."

  OK para tentar novamente: não.

CreateMLTaskRunActivity

Essa atividade tem as seguintes exceções:

- InvalidInputException (400)
  - "Falha de serviço interno devido a entrada inesperada."
  - “Uma fonte de entrada da tabela do AWS Glue deve ser especificada na transformação.”
  - "A coluna de fonte de entrada [columnName] tem um tipo de dados inválido definido no catálogo."
  - "Exatamente uma tabela de registro de entrada deve ser fornecida."
  - "Deve especificar o nome do banco de dados."
  - "Deve especificar o nome da tabela."
  - "O esquema não está definido na transformação."
  - "O esquema deve conter a chave primária fornecida: [primaryKey]."
  - "Problema ao obter o esquema do catálogo de dados: [message]."
  - "Não é possível definir a capacidade máxima e o Núm./Tipo de operador ao mesmo tempo."

- "Tanto WorkerType quanto NumberOfWorkers devem ser definidos."
- "MaxCapacity deve ser >= [maxCapacity]."
- "NumberOfWorkers deve ser >= [maxCapacity]."
- "O máximo de novas tentativas não deve ser negativo."
- "Os parâmetros de Encontrar correspondências não foram definidos."
- "Uma chave primária deve ser especificada nos parâmetros de Encontrar correspondências."

OK para tentar novamente: não.

- AlreadyExistsException (400)
  - "Já existe uma transformação com o nome [transformName]."

  OK para tentar novamente: não.
- IdempotentParameterMismatchException (400)
  - "A solicitação de criação idempotente para a transformação [transformName] tinha parâmetros não correspondentes."

  OK para tentar novamente: não.
- InternalServiceException (500)
  - "Falha de dependência."

  OK para tentar novamente: sim.
- ResourceNumberLimitExceededException (400)
  - "A contagem de transformações de ML ([count]) excedeu o limite de [limit] transformações."

  OK para tentar novamente: sim, depois de excluir uma transformação a fim de criar espaço para a nova.

DeleteMLTransformActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]"

  OK para tentar novamente: não.

GetMLTaskRunActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."
  - "Nenhuma execução de tarefa de ML encontrada para [taskRunId]: na conta [accountId] para a transformação [transformName]."

  OK para tentar novamente: não.

GetMLTaskRunsActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."
  - "Nenhuma execução de tarefa de ML encontrada para [taskRunId]: na conta [accountId] para a transformação [transformName]."

  OK para tentar novamente: não.

GetMLTransformActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."

  OK para tentar novamente: não.

GetMLTransformsActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."

  OK para tentar novamente: não.
- InvalidInputException (400)
  - "O ID da conta não pode ficar em branco."
  - "Classificação não compatível com a coluna [column]."
  - "[column] não pode ficar em branco."
  - "Falha de serviço interno devido a entrada inesperada."

  OK para tentar novamente: não.

GetSaveLocationForTransformArtifactActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."

  OK para tentar novamente: não.
- InvalidInputException (400)
  - "Tipo de artefato [artifactType] não compatível."
  - "Falha de serviço interno devido a entrada inesperada."

  OK para tentar novamente: não.

GetTaskRunArtifactActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."

- "Nenhuma execução de tarefa de ML encontrada para [taskRunId]: na conta [accountId] para a transformação [transformName].”

OK para tentar novamente: não.

- InvalidInputException (400)
  - "O nome de arquivo '[fileName]' é inválido para publicação."
  - "Não é possível recuperar o artefato para o tipo de tarefa [taskType]."
  - "Não é possível recuperar o artefato para [artifactType]."
  - "Falha de serviço interno devido a entrada inesperada."

OK para tentar novamente: não.

PublishMLTransformModelActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."
  - "Não foi possível encontrar um modelo existente com a versão – [version] para a conta de ID – [accountId] – e ID de transformação – [transformId]."

OK para tentar novamente: não.

- InvalidInputException (400)
  - "O nome de arquivo '[fileName]' é inválido para publicação."
  - "Sinal de menos inicial ilegal na string não assinada [string]."
  - "Dígito inválido no final de [string]."
  - "O valor da string [string] excede o intervalo de longo não assinado."
  - "Falha de serviço interno devido a entrada inesperada."

OK para tentar novamente: não.

PullLatestMLTransformModelActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."

  OK para tentar novamente: não.
- InvalidInputException (400)
  - "Falha de serviço interno devido a entrada inesperada."

  OK para tentar novamente: não.
- ConcurrentModificationException (400)
  - "Não é possível criar a versão do modelo para treinar devido a inserções de corrida com parâmetros incompatíveis."
  - "O modelo de transformação de ML para transformação de ID [transformId] está obsoleto ou está sendo atualizado por outro processo. Tente novamente."

  OK para tentar novamente: sim.

PutJobMetadataForMLTransformActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."
  - "Nenhuma execução de tarefa de ML encontrada para [taskRunId]: na conta [accountId] para a transformação [transformName]."

  OK para tentar novamente: não.
- InvalidInputException (400)
  - "Falha de serviço interno devido a entrada inesperada."
  - "Tipo de metadados de trabalho [jobType] desconhecido."
  - "É necessário fornecer um ID de execução de tarefa para atualizar."

  OK para tentar novamente: não.

StartExportLabelsTaskRunActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."
  - "Não existe nenhum conjunto de rótulos para transformId [transformId] na conta de ID [accountId]."

  OK para tentar novamente: não.
- InvalidInputException (400)
  - "[message]."
  - "O caminho do S3 fornecido não está na mesma região que a transformação. Esperava-se a região – [region], mas foi obtida – [region]."

  OK para tentar novamente: não.

StartImportLabelsTaskRunActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."

  OK para tentar novamente: não.
- InvalidInputException (400)
  - "[message]."
  - "Caminho de arquivo de rótulo inválido."
  - "Não é possível acessar o arquivo de rótulo em [labelPath]. [message]."
  - "Não é possível usar a função do IAM fornecida na transformação. Função: [role]."
  - "Arquivo de rótulo inválido de tamanho 0."
  - "O caminho do S3 fornecido não está na mesma região que a transformação. Esperava-se a região – [region], mas foi obtida – [region]."

OK para tentar novamente: não.

- ResourceNumberLimitExceededException (400)
  - "O arquivo de rótulo excedeu o limite de [limit] MB."

  OK para tentar novamente: não. Considere dividir o arquivo de rótulo em vários arquivos menores.

StartMLEvaluationTaskRunActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."

  OK para tentar novamente: não.
- InvalidInputException (400)
  - "Exatamente uma tabela de registro de entrada deve ser fornecida."
  - "Deve especificar o nome do banco de dados."
  - "Deve especificar o nome da tabela."
  - "Os parâmetros de Encontrar correspondências não foram definidos."
  - "Uma chave primária deve ser especificada nos parâmetros de Encontrar correspondências."

  OK para tentar novamente: não.
- MLTransformNotReadyException (400)
  - "Esta operação só pode ser aplicada a uma transformação que está em um estado READY."

  OK para tentar novamente: não.
- InternalServiceException (500)
  - "Falha de dependência."

  OK para tentar novamente: sim.
- ConcurrentRunsExceededException (400)
  - "A contagem de execuções de tarefas de ML [count] excedeu o limite de transformação de [limit] execuções de tarefas."

- "A contagem de execuções de tarefas de ML [count] excedeu o limite de [limit] execuções de tarefas."

OK para repetir: sim, depois de aguardar a conclusão das execuções da tarefa.

StartMLLabelingSetGenerationTaskRunActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."

  OK para tentar novamente: não.
- InvalidInputException (400)
  - "Exatamente uma tabela de registro de entrada deve ser fornecida."
  - "Deve especificar o nome do banco de dados."
  - "Deve especificar o nome da tabela."
  - "Os parâmetros de Encontrar correspondências não foram definidos."
  - "Uma chave primária deve ser especificada nos parâmetros de Encontrar correspondências."

  OK para tentar novamente: não.
- InternalServiceException (500)
  - "Falha de dependência."

  OK para tentar novamente: sim.
- ConcurrentRunsExceededException (400)
  - "A contagem de execuções de tarefas de ML [count] excedeu o limite de transformação de [limit] execuções de tarefas."

  OK para repetir: sim, após a conclusão das execuções de tarefa.

UpdateMLTransformActivity

Essa atividade tem as seguintes exceções:

- EntityNotFoundException (400)
  - "Não é possível encontrar MLTransform na conta [accountId] com o identificador [transformName]."

  OK para tentar novamente: não.
- InvalidInputException (400)
  - "Já existe outra transformação com o nome [transformName]."
  - "[message]."
  - "O nome da transformação não pode ficar em branco."
  - "Não é possível definir a capacidade máxima e o Núm./Tipo de operador ao mesmo tempo."
  - "Tanto WorkerType quanto NumberOfWorkers devem ser definidos."
  - "MaxCapacity deve ser >= [minMaxCapacity]."
  - "NumberOfWorkers deve ser >= [minNumWorkers]."
  - "O máximo de novas tentativas não deve ser negativo."
  - "Falha de serviço interno devido a entrada inesperada."
  - "Os parâmetros de Encontrar correspondências não foram definidos."
  - "Uma chave primária deve ser especificada nos parâmetros de Encontrar correspondências."

  OK para tentar novamente: não.
- AlreadyExistsException (400)
  - "Já existe uma transformação com o nome [transformName]."

  OK para tentar novamente: não.
- IdempotentParameterMismatchException (400)
  - "A solicitação de criação idempotente para a transformação [transformName] tinha parâmetros não correspondentes."

  OK para tentar novamente: não.

Cotas do AWS Glue

Você pode entrar em contato com o AWS Support para solicitar um aumento de quota para as quotas de serviço listadas na Referência geral da AWS. A menos que especificado de outra forma,

cada cota é específica da região . Para obter mais informações, consulte Endpoints e cotas do AWS Glue.

Melhorando AWS Glue o desempenho

Estratégia de linha de base para ajuste de performance

Para melhorar o AWS Glue desempenho, você pode considerar a atualização de determinados AWS Glue parâmetros relacionados ao desempenho. Quando for ajustar os parâmetros, observe as seguintes práticas recomendadas:

- Determine suas metas de performance antes de começar a identificar os problemas.
- Use as métricas para identificar os problemas antes de tentar alterar os parâmetros de ajuste.

Para obter os resultados mais consistentes ao ajustar um trabalho, desenvolva uma estratégia de linha de base para fazer os ajustes.

Geralmente, o ajuste de performance é feito no seguinte fluxo de trabalho:

1. Determine as metas de performance.
2. Meça as métricas.
3. Identifique os gargalos.
4. Reduza o impacto dos gargalos.
5. Repita as etapas de 2 a 4 até atingir a meta pretendida.

Estratégias de ajuste para seu tipo de trabalho

Tarefas do Spark — siga as orientações em Práticas recomendadas AWS Glue para ajuste de desempenho das tarefas do Apache Spark em Orientação prescritiva. AWS

Outros trabalhos — você pode ajustar os trabalhos AWS Glue de shell do Ray e do AWS Glue Python adaptando as estratégias disponíveis em outros ambientes de tempo de execução.

Melhorar a performance do AWS Glue em trabalhos do Apache Spark

Para melhorar a performance do AWS Glue for Spark, você pode considerar a atualização de determinados parâmetros de performance do AWS Glue e do Spark.

Para obter mais informações sobre estratégias específicas para identificar gargalos por meio de métricas e reduzir seu impacto, consulte Best practices for performance tuning AWS Glue for Apache Spark jobs em AWS Prescriptive Guidance. Este guia apresenta os principais tópicos aplicáveis ao Apache Spark em todos os ambientes de runtime, como a arquitetura do Spark e conjuntos de dados distribuídos resilientes. Usando esses tópicos, o guia orienta você a implementar estratégias específicas de ajuste de performance, como otimizar embaralhamentos e paralelizar tarefas.

É possível identificar gargalos configurando AWS Glue para mostrar a interface do usuário do Spark. Para ter mais informações, consulte the section called “Monitorar com a interface do usuário do Spark”.

Além disso, o AWS Glue fornece recursos de performance que podem ser aplicáveis ao tipo específico de datastore ao qual seu trabalho se conecta. Informações de referência sobre parâmetros de performance para armazenamentos de dados podem ser encontradas em the section called “Parâmetros de conexão”.

Otimizando leituras com o pushdown no Glue ETL AWS

Pushdown é uma técnica de otimização que aproxima a lógica de recuperação de dados da fonte de seus dados. A fonte pode ser um banco de dados ou um sistema de arquivos, como o Amazon S3. Ao executar determinadas operações diretamente na fonte, você pode economizar tempo e poder de processamento não levando todos os dados da rede para o mecanismo Spark gerenciado pelo AWS Glue.

Em outras palavras, o pushdown reduz as varreduras de dados. Para obter mais informações sobre o processo para identificar quando essa técnica é adequada, consulte Reduce the amount of data scan no guia Best practices for performance tuning AWS Glue for Apache Spark jobs em AWS Prescriptive Guidance.

Pushdown de predicados em arquivos armazenados no Amazon S3

Ao trabalhar com arquivos no Amazon S3 que foram organizados por prefixo, você pode filtrar os caminhos de destino do Amazon S3 definindo um predicado de pushdown. Em vez de ler o conjunto de dados completo e aplicar filtros em um `DynamicFrame`, você pode aplicar o filtro diretamente aos metadados da partição armazenados no catálogo de dados do AWS Glue. Essa abordagem permite listar e ler seletivamente somente os dados necessários. Para obter mais informações sobre esse processo, incluindo gravação em um bucket por partições, consulte the section called “Gerenciar partições”.

Você consegue fazer pushdown de predicados no Amazon S3 usando o parâmetro `push_down_predicate`. Considere um bucket no Amazon S3 que você particionou por ano, mês e dia. Se você quiser recuperar dados de clientes de junho de 2022, poderá instruir o AWS Glue a ler somente os caminhos relevantes do Amazon S3. Nesse caso, `push_down_predicate` é `year='2022' and month='06'`. Resumindo, a operação de leitura pode ser realizada conforme abaixo:

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()
```

No cenário anterior, `push_down_predicate` recupera uma lista de todas as partições do catálogo de dados do AWS Glue e as filtra antes de ler os arquivos subjacentes do Amazon S3. Embora isso ajude na maioria dos casos, ao trabalhar com conjuntos de dados que têm milhões de partições, o processo de listar partições pode ser demorado. Para resolver esse problema, a remoção de partições do lado do servidor pode ser usada para melhorar a performance. Isso é feito criando um índice de partição para seus dados no catálogo de dados do AWS Glue. Para obter mais informações sobre particionamento, consulte the section called “Trabalhar com índices de partição”. Depois, você pode usar a opção `catalogPartitionPredicate` para referenciar o índice. Para obter um exemplo de recuperação de partições com `catalogPartitionPredicate`, consulte the section called “Predicados de partição de catálogo”.

Pushdown ao trabalhar com fontes JDBC

O leitor de JDBC do AWS Glue usado no `GlueContext` é compatível com pushdown em bancos de dados compatíveis, fornecendo consultas SQL personalizadas que podem ser executadas

diretamente na fonte. Isso pode ser feito definindo o parâmetro `sampleQuery`. Sua consulta de amostra pode especificar quais colunas selecionar, bem como fornecer um predicado de pushdown para limitar os dados transferidos para o mecanismo Spark.

Por padrão, as consultas de amostra operam em um único nó, o que pode resultar em falhas de trabalho ao lidar com grandes volumes de dados. Para usar esse atributo para consultar dados em grande escala, você deve configurar o particionamento de consultas definindo `enablePartitioningForSampleQuery` como verdadeiro, o que distribuirá a consulta para vários nós em uma chave de sua escolha. O particionamento de consultas também requer alguns outros parâmetros de configuração necessários. Para obter mais informações sobre particionamento de consultas, consulte the section called “Leitura do JDBC em paralelo”.

Ao configurar `enablePartitioningForSampleQuery`, o AWS Glue combinará seu predicado de pushdown com um predicado de particionamento ao consultar seu banco de dados. A `sampleQuery` deve terminar com um `AND` para o AWS Glue anexar as condições de particionamento. (Se você não fornecer um predicado de pushdown, a `sampleQuery` deverá terminar com um `WHERE`). Veja um exemplo abaixo, em que fizemos o pushdown de um predicado para recuperar somente linhas com `id` maiores do que 1000. Essa `sampleQuery` retornará somente as colunas de nome e localização das linhas em que `id` for maior que o 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**
>
> Se `customer_tbl` tiver um nome diferente no Catálogo de Dados e no datastore subjacente, você deverá fornecer o nome da tabela subjacente em sample_query, pois a consulta é transmitida para o datastore subjacente.

Você também pode consultar tabelas JDBC sem fazer a integração com o catálogo de dados do AWS Glue. Em vez de fornecer o nome de usuário e a senha como parâmetros para o método, você pode reutilizar as credenciais de uma conexão preexistente fornecendo e `useConnectionProperties` e `connectionName`. Neste exemplo, recuperamos as credenciais a partir de uma conexão denominada `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 e limitações para pushdown no AWS Glue

Pushdown, como conceito, é aplicável ao ler de fontes que não são de streaming. O AWS Glue é compatível com uma variedade de fontes; a capacidade de pushdown depende da fonte e do conector.

- Ao se conectar ao Snowflake, você pode usar a opção `query`. Existe uma funcionalidade semelhante no conector do Redshift no AWS Glue 4.0 e versões posteriores. Para obter mais informações sobre como doe Snowflake com `query`, consulte the section called “Ler no Snowflake”.
- O leitor de ETL do DynamoDB não é compatível com filtros ou predicados de aplicação. O MongoDB e o DocumentDB também não são compatíveis com esse tipo de funcionalidade.
- Ao ler dados armazenados do Amazon S3 em formatos de tabela aberta, o método de particionamento para arquivos no Amazon S3 não é mais suficiente. Para ler e gravar de partições usando formatos de tabela aberta, consulte a documentação do formato.
- Os métodos DynamicFrame não executam o pushdown de projeção do Amazon S3. Todas as colunas serão lidas nos arquivos que passam pelo filtro de predicados.

- Ao trabalhar com conectores `custom.jdbc` no AWS Glue, a capacidade de executar o pushdown depende da fonte e do conector. Consulte a documentação apropriada do conector para confirmar se e como ele é compatível com pushdown no AWS Glue.

Usar Auto Scaling para o AWS Glue

Agora o Auto Scaling está disponível para seus trabalhos de ETL e transmissão do AWS Glue com o AWS Glue versão 3.0 ou posterior.

Com o Auto Scaling habilitado, você obtém os seguintes benefícios:

- O AWS Glue adiciona e remove automaticamente operadores do cluster, dependendo do paralelismo em cada estágio ou microlote da execução do trabalho.
- Ele elimina a necessidade de experimentar e decidir quantos operadores deverão ser designados para os trabalhos de ETL do AWS Glue.
- Se você escolher o número máximo de operadores, o AWS Glue escolherá os recursos de tamanho certo para a workload.
- Você pode ver como o tamanho do cluster se altera durante a execução do trabalho observando as métricas do CloudWatch na página de detalhes da execução de trabalho no AWS Glue Studio.

O Auto Scaling para trabalhos de ETL e streaming do AWS Glue permite aumentar e reduzir verticalmente a escala dos recursos de computação dos trabalhos do AWS Glue sob demanda. Aumentar a escala verticalmente sob demanda ajuda você a alocar apenas os recursos de computação necessários ao iniciar a execução do trabalho e também a provisionar os recursos necessários de acordo com a demanda durante o trabalho.

O Auto Scaling também é compatível com a redução dinâmica da escala vertical dos recursos do trabalho do AWS Glue durante sua execução. Em uma execução de trabalho, quando mais executores forem solicitados pela aplicação Spark, mais operadores serão adicionados ao cluster. Quando o executor ficar ocioso sem tarefas de computação ativas, o executor e o operador correspondente serão removidos.

Cenários comuns em que o Auto Scaling ajuda com o custo e a utilização de suas aplicações do Spark incluem um driver do Spark listando um grande número de arquivos no Amazon S3 ou executando uma carga enquanto os executores estão inativos, estágios do Spark em execução com apenas alguns executores devido ao excesso de provisionamento e distorções de dados ou demanda de computação irregular nos estágios do Spark.

Requisitos

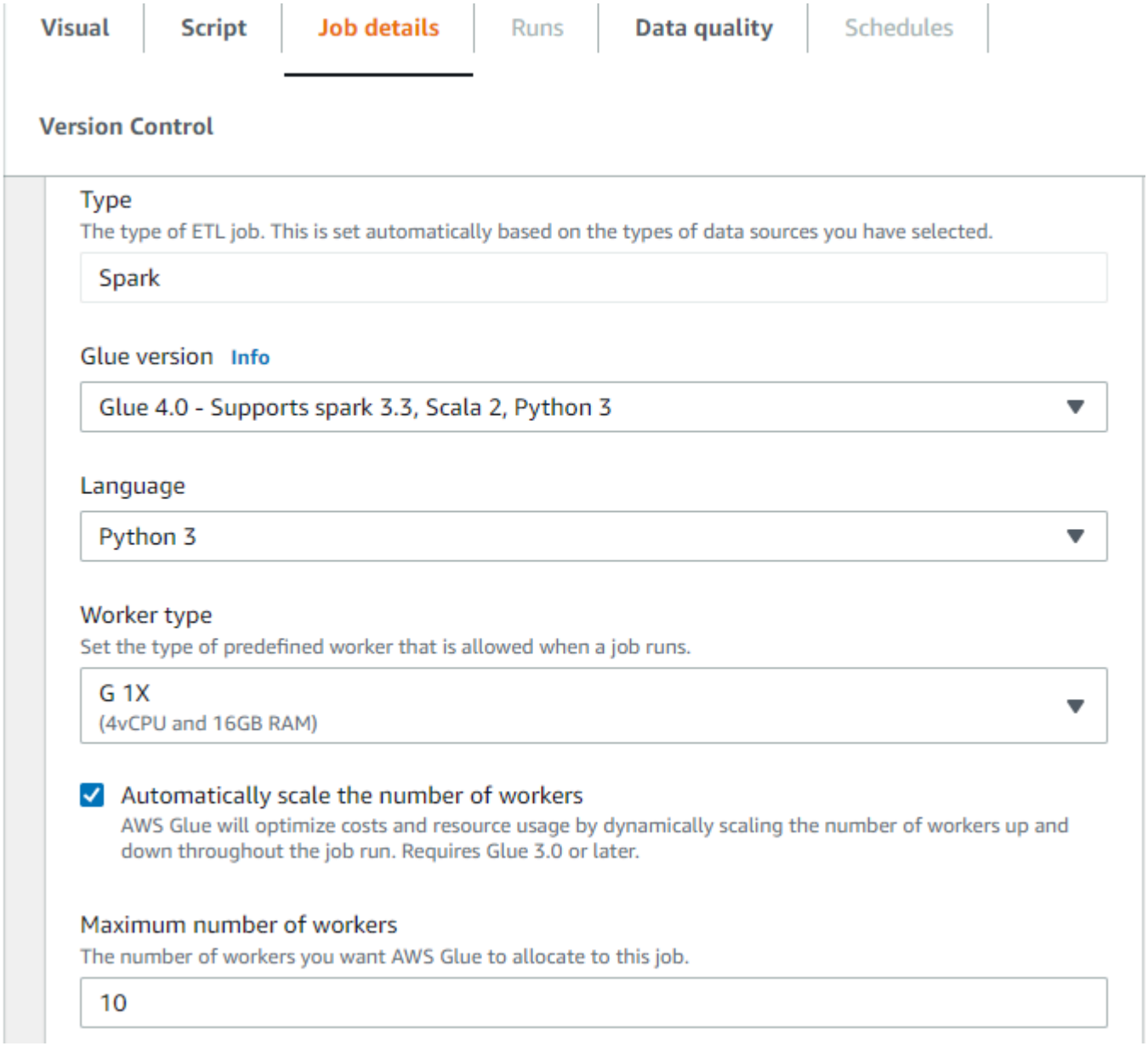
O Auto Scaling só está disponível para o AWS Glue versão 3.0 ou posterior. Para usar o Auto Scaling, você pode seguir o guia de migração para migrar os trabalhos existentes para o AWS Glue versão 3.0 ou criar novos trabalhos com o AWS Glue versão 3.0.

O ajuste de escala automático está disponível para trabalhos AWS Glue com os tipos de trabalho `G.1X`, `G.2X`, `G.4X`, `G.8X` ou `G.025X` (somente para trabalhos de streaming). Não há suporte para DPUs padrão.

Habilitar Auto Scaling no AWS Glue Studio

Na guia Job details (Detalhes do trabalho) no AWS Glue Studio, escolha o tipo Spark ou Spark Streaming (Streaming do Spark) e a Glue version (Versão do Glue) como **`Glue 3.0`** ou **`Glue 4.0`**. Uma caixa de seleção será exibida abaixo de Worker type (Tipo de operador).

- Selecione a opção Automatically scale the number of workers (Dimensionar automaticamente o número de operadores).
- Defina Maximum numer of workers (Número máximo de operadores) para estabelecer o número máximo de operadores que podem ser transferidos para a execução do trabalho.



Habilitar o Auto Scaling com a AWS CLI ou SDK

Para habilitar o Auto Scaling diretamente da AWS CLI para sua execução de trabalho, execute `start-job-run` com a seguinte configuração:

```
{
    "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...
}
```

Quando a execução do trabalho de ETL estiver concluída, também será possível chamar `get-job-run` para verificar o uso efetivo de recurso da execução de trabalho em segundos de DPU. Observação: o novo campo DPUSeconds só aparecerá para trabalhos em lote no AWS Glue 3.0 ou posterior habilitado com o Auto Scaling. Esse campo não é compatível com trabalhos de transmissão.

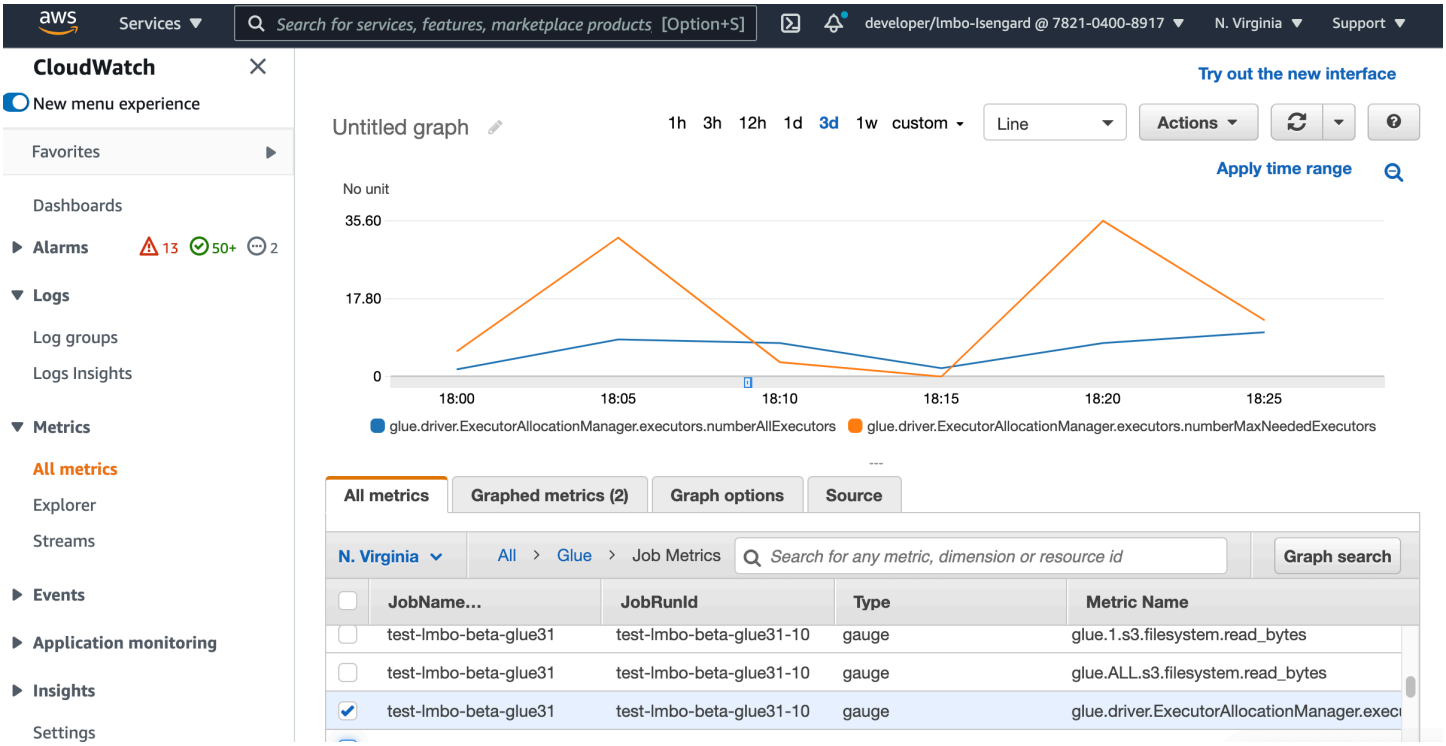
```
$ aws glue get-job-run --job-name your-job-name --run-id jr_xx --endpoint https://
glue.us-east-1.amazonaws.com --region us-east-1
{
    "JobRun": {
        ...
        "GlueVersion": "3.0",
        "DPUSeconds": 386.0
    }
}
```

Você também pode configurar execuções de tarefas com o Auto Scaling usando o AWS Glue SDK com a mesma configuração.

Monitorar o Auto Scaling com métricas do Amazon CloudWatch

As métricas do executor do CloudWatch estarão disponíveis para os trabalhos do AWS Glue 3.0 se você habilitar o Auto Scaling. As métricas podem ser usadas para monitorar a demanda e o uso otimizado dos executores em suas aplicações do Spark habilitadas com o Auto Scaling. Para ter mais informações, consulte Monitorar o AWS Glue usando métricas do Amazon CloudWatch.

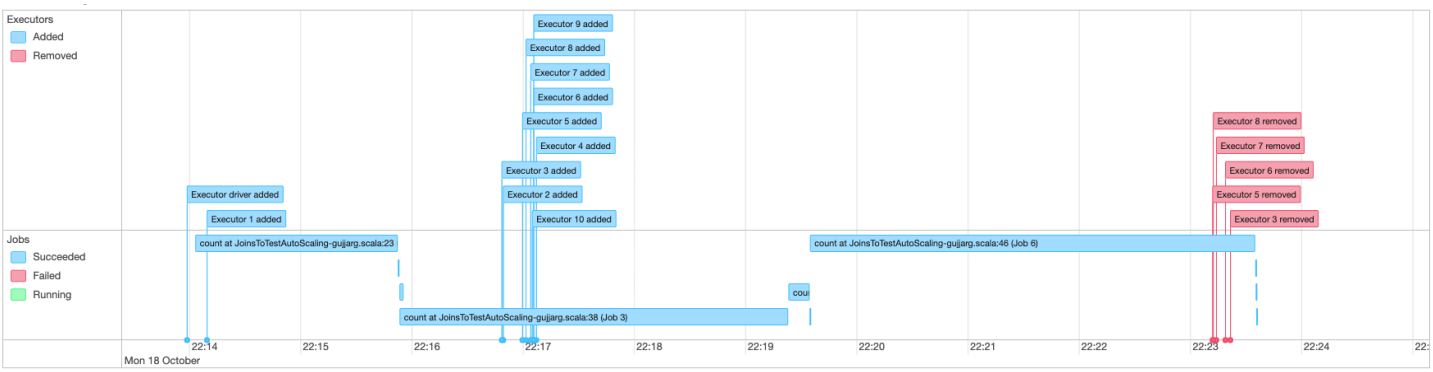
- glue.driver.ExecutorAllocationManager.executors.numberAllExecutors
- glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors



Para obter mais detalhes sobre essas métricas, consulte [Monitorar planejamento de capacidade de DPU](#).

Monitoramento do Auto Scaling com o Spark UI

Com o Auto Scaling habilitado, você também pode monitorar os executores que estão sendo adicionados e removidos com o aumento e a redução dinâmicos da escala vertical de acordo com a demanda dos trabalhos do AWS Glue usando a interface de usuário do Spark no Glue. Para ter mais informações, consulte [Habilitar a interface do usuário da Web do Apache Spark para trabalhos do AWS Glue](#).



Monitoramento do uso de DPU da execução de trabalho do Auto Scaling

É possível usar a visualização de execução de trabalhos do AWS Glue Studio para verificar o uso de DPU dos trabalhos de ajuste de escala automático.

1. Escolha Monitoramento no painel de navegação do AWS Glue Studio. A página de monitoramento será exibida.
2. Localize o quadro Job runs (Execuções de trabalho).
3. Acesse a execução de trabalho de seu interesse e localize a coluna DPU hours (Horas de DPU) para verificar o uso da execução de trabalho específica.

Limitações

O Auto Scaling de streaming do AWS Glue no momento não é compatível com a união de um DataFrame de streaming com um DataFrame estático criado fora do `ForEachBatch`. Um DataFrame estático criado dentro do `ForEachBatch` funcionará conforme esperado.

Particionar workloads com execução limitada

Erros em aplicações do Spark geralmente surgem de scripts ineficientes do Spark, execução distribuída na memória de transformações em grande escala e anormalidades do conjunto de dados. Há muitas razões que podem causar problemas de memória de driver ou executor, por exemplo, uma distorção de dados, listar muitos objetos ou colocar em ordem aleatória dados grandes. Esses problemas geralmente aparecem quando você está processando grandes quantidades de dados de backlog com o Spark.

O AWS Glue permite que você resolva problemas de OOM e torne seu processamento ETL mais fácil com o particionamento de workload. Com o particionamento de workload habiltado, cada execução de trabalho de ETL seleciona somente dados não processados, com um limite superior no tamanho do conjunto de dados ou o número de arquivos a serem processados com essa execução de trabalho. Futuras execuções de trabalho processarão os dados restantes. Por exemplo, se houver 1.000 arquivos que precisem ser processados, você pode definir o número de arquivos para 500 e separá-los em duas execuções de trabalho.

O particionamento de workload é suportado apenas para fontes de dados do Amazon S3.

Habilitar o particionamento de workloads

Você pode habilitar a execução limitada definindo manualmente as opções no seu script ou adicionando propriedades da tabela de catálogo.

Para habilitar o particionamento de workload com execução limitada no seu script:

1. Para evitar o reprocessamento de dados, habilite os marcadores de trabalho no novo trabalho ou no já existente. Para obter mais informações, consulte Rastreamento de dados processados usando marcadores de trabalho.
2. Modifique seu script e defina o limite nas opções adicionais na API `getSource` do AWS Glue. Você também deve definir o contexto de transformação para que o marcador de trabalho armazene o elemento `state`. Por exemplo:

   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 o particionamento de workload com execução limitada na tabela do Data Catalog:

1. Defina os pares de chave-valor no campo `parameters` de sua estrutura de tabela no Data Catalog. Para obter mais informações, consulte Visualização e edição dos detalhes da tabela.
2. Defina o limite superior para o tamanho do conjunto de dados ou o número de arquivos processados:

   - Defina `boundedSize` para o tamanho de destino do conjunto de dados em bytes. A execução do trabalho será interrompida depois de atingir o tamanho desejado da tabela.
   - Defina `boundedFiles` para o número de destino de arquivos. A execução do trabalho será interrompida após o processamento do número de arquivos desejado.

> **Note**
>
> Você deve definir apenas um entre `boundedSize` ou `boundedFiles`, pois somente um limite é suportado.

Configurar um acionador do AWS Glue para executar o trabalho automaticamente

Depois de habilitar a execução limitada, é possível configurar um acionador do AWS Glue para executar automaticamente o trabalho e carregar os dados incrementalmente em execuções sequenciais. Vá para console do AWS Glue e crie um acionador, configure o horário de programação e anexe-o ao seu trabalho. Em seguida, ele acionará automaticamente a próxima execução do trabalho e processará o novo lote de dados.

Você também pode usar fluxos de trabalho do AWS Glue para orquestrar vários trabalhos a fim de processar dados de diferentes partições simultaneamente. Para obter mais informações, consulte Acionadores do AWS Glue e Fluxos de trabalho do AWS Glue.

Para obter mais informações sobre casos de uso e opções, consulte o blog Optimizing Spark applications with workload partitioning in AWS Glue.

Problemas conhecidos do AWS Glue

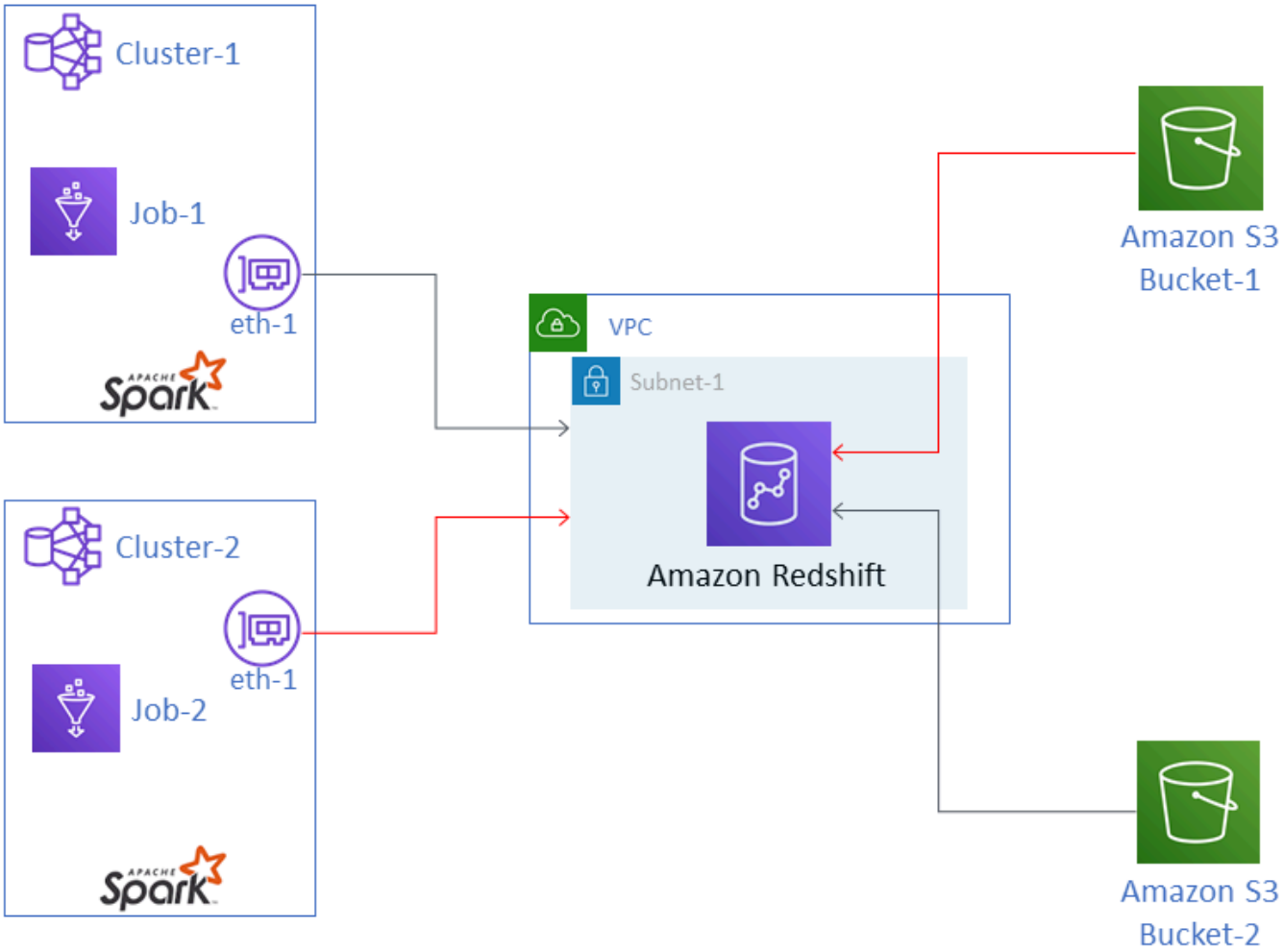
Observe os seguintes problemas conhecidos do AWS Glue.

Tópicos

Impedir acesso a dados entre trabalhos

Considere a situação em que você tenha dois trabalhos do AWS Glue Spark em uma única conta da AWS, cada um em execução em um cluster do AWS Glue Spark separado. Os trabalhos estão usando conexões do AWS Glue para acessar recursos na mesma nuvem privada virtual (VPC). Nessa situação, um trabalho em execução em um cluster pode acessar os dados do trabalho em execução no outro cluster.

O diagrama a seguir ilustra um exemplo dessa situação.



No diagrama, o AWS Glue `Job-1` está em execução no `Cluster-1`, e o Job-2 está em execução no `Cluster-2`. Os dois trabalhos estão atuando com a mesma instância do Amazon RedShift, que reside na `Subnet-1` de uma VPC. A `Subnet-1` pode ser uma sub-rede pública ou privada.

`Job-1` está transformando dados do `Bucket-1` do Amazon Simple Storage Service (Amazon S3) e gravando-os no Amazon RedShift. `Job-2` está fazendo o mesmo com os dados do `Bucket-2`. `Job-1` usa a função do AWS Identity and Access Management (IAM) `Role-1` (não mostrada), que dá acesso ao `Bucket-1`. `Job-2` usa `Role-2` (não mostrada), que dá acesso ao `Bucket-2`.

Esses trabalhos têm caminhos de rede que permitem que eles se comuniquem com os clusters uns dos outros e, assim, acessem os dados uns dos outros. Por exemplo, o `Job-2` pode acessar dados no `Bucket-1`. No diagrama, isso é mostrado como o caminho em vermelho.

Para evitar essa situação, recomendamos que você associe diferentes configurações de segurança ao `Job-1` e ao `Job-2`. Ao associar as configurações de segurança, o acesso entre trabalhos aos dados é bloqueado em virtude dos certificados que o AWS Glue cria. As configurações de segurança podem ser fictícias. Ou seja, é possível criar as configurações de segurança sem habilitar a criptografia de dados do Amazon S3, dados do Amazon CloudWatch ou marcadores de trabalho. Todas as três opções de criptografia podem ser desabilitadas.

Para obter mais informações sobre configurações de segurança, consulte the section called “Criptografar dados gravados pelo AWS Glue”.

Como associar uma configuração de segurança a um trabalho

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Na página Configure the job properties (Configurar as propriedades do trabalho) para o trabalho, expanda a seção Security configuration, scripts libraries e job parameters (Configuração de segurança, bibliotecas de scripts e parâmetros de trabalho).
3. Selecione uma configuração de segurança na lista.

Histórico de documentação do AWS Glue

Alteração	Descrição	Data
Support para perfis AWS Glue de uso	Os administradores podem criar perfis de AWS Glue uso para várias classes de usuários na conta, como desenvolvedores, testadores e equipes de produtos. Essa flexibilidade permite que os administradores apliquem diferentes controles de uso e custo para cada classe de usuários. Para obter mais informações, consulte Configuração de perfis AWS Glue de uso.	18 de junho de 2024
Support para um conector Salesforce para AWS Glue Spark	Foram adicionadas informaçõ es sobre um novo AWS Glue conector para o Salesforc e. Esse recurso permite que você use o Spark AWS Glue para ler e gravar no Salesforc e nas AWS Glue versões 4.0 e posteriores. Para obter mais informações, consulte Conectando-se ao Salesforce.	22 de maio de 2024
Integração de dados do Amazon Q em AWS Glue (GA)	A integração de dados do Amazon Q AWS Glue é um novo recurso generativ o de IA AWS Glue que permite que engenheiros de dados e desenvolvedores	30 de abril de 2024

	de ETL criem trabalhos de integração de dados usando linguagem natural. Engenheiros e desenvolvedores podem pedir a Q que crie trabalhos, solucione problemas e responda perguntas sobre AWS Glue integração de dados. Para obter mais informações, consulte Integração de dados do Amazon Q no AWS Glue. Esse recurso inclui uma atualização do `AwsGlueSessionUserRestrictedPolicy` , `AwsGlueSessionUserRestrictedNotebookServiceRole` , e políticas `AwsGlueSessionUserRestrictedServiceRole` AWS gerenciadas. Para obter mais informações, consulte AWS Glue atualizações nas políticas AWS gerenciadas.

Integração de dados do Amazon Q em AWS Glue (versão prévia)	A integração de dados do Amazon Q AWS Glue é um novo recurso generativo de IA AWS Glue que permite que engenheiros de dados e desenvolvedores de ETL criem trabalhos de integração de dados usando linguagem natural. Engenheiros e desenvolvedores podem pedir a Q que crie trabalhos, solucione problemas e responda perguntas sobre AWS Glue integração de dados. Para obter mais informações, consulte Integração de dados do Amazon Q no AWS Glue. Esse recurso inclui uma atualização da política `AwsGlueSessionUserRestrictedNotebookPolicy` AWS gerenciada. Para obter mais informações, consulte AWS Glue atualizações nas políticas AWS gerenciadas.	30 de janeiro de 2024

Atualização da documentação do AWS Glue Streaming	Foi adicionado um novo capítulo com conteúdo novo e reorganizado para AWS Glue streaming. Este conteúdo descreve como o streaming funciona AWS Glue, as características do processam ento de dados em tempo real e como monitorar suas tarefas de streaming. Para obter mais informações, consulte AWS Glue Streaming.	27 de dezembro de 2023
Suporte para a detecção detalhada de dados confidenc iais	A transformação “Detectar dados confidenciais” permite detectar, mascarar ou remover entidades definidas por você ou que são predefinidas pelo AWS Glue. Além disso, ações minuciosas permitem que você aplique uma ação específica por entidade. Para obter mais informaçõ es, consulte Como usar a detecção detalhada de dados confidenciais .	26 de novembro de 2023

Support para monitorar trabalhos com métricas de AWS Glue observabilidade	Use métricas de observabilidade do AWS Glue para gerar insights sobre o que está acontecendo dentro do seu AWS Glue para trabalhos do Apache Sparks para melhorar a triagem e a análise de problemas. Para obter mais informações, consulte Monitoramento com métricas e observabilidade do AWS Glue.	26 de novembro de 2023
Support para detecção de anomalias na qualidade AWS Glue de dados	A detecção de anomalias no AWS Glue Data Quality usa algoritmos de machine learning (ML) nas estatísticas de dados ao longo do tempo para detectar padrões anormais e problemas ocultos de qualidade de dados que são difíceis de detectar por meio de regras. Para obter mais informações, consulte Detecções de anomalias no AWS Glue Data Quality .	26 de novembro de 2023

Atualização do comportamento do registro em log padrão da IU do Spark	Os trabalhos do Spark que geram registros da interface do Spark agora serão gravados com um padrão de nome de arquivo diferente para suportar a interface do Spark no console. AWS Glue Isso não altera o comportamento do CloudWatch log. É possível reverter para o comportamento antigo atualizando a configuração do seu trabalho. Para obter mais informações, consulte Como monitorar os trabalhos usando a IU web do Spark.	17 de novembro de 2023

Support para novas fontes de dados no AWS Glue Spark	As conexões com o Amazon OpenSearch Service, o Azure SQL, o Azure Cosmos for NoSQL, o SAP HANA, o Teradata Vantage e o Vertica agora têm suporte nativo. AWS Glue Além disso, as conexões com essas fontes de dados, junto com o MongoDB, agora estão disponíveis para uso no editor visual AWS Glue do Studio. Para obter mais informações, consulte Tipos e opções de conexão para ETL no Spark AWS Glue para obter informações sobre AWS Glue o suporte ao Spark e Adicionar uma AWS Glue conexão para obter informaçõ es sobre o uso no editor visual do AWS Glue Studio.	17 de novembro de 2023
Suporte à geração de estatísti cas de colunas	Você pode calcular estatísti cas em nível de coluna para AWS Glue Data Catalog tabelas em formatos de dados como Parquet, ORC, JSON, ION, CSV e XML sem configurar pipelines de dados adicionais. Para obter mais informações, consulte Trabalhar com estatísticas de colunas.	16 de novembro de 2023

Suporte à compactação de dados para tabelas do Iceberg	Para melhor desempenho de leitura por serviços de AWS análise, como Amazon Athena e Amazon EMR, e trabalhos de AWS Glue ETL, o Data Catalog fornece compactação gerenciada (um processo que compacta pequenos objetos do Amazon S3 em objetos maiores) para tabelas Iceberg no Data Catalog. Para obter mais informações, consulte Como otimizar tabelas do Iceberg.	13 de novembro de 2023
Atualização do comportamento de espera na execução do trabalho	As execuções de trabalhos de shell padrão do Spark e do Python agora avançarão para `WAITING` em determina das situações em vez de avançarem imediatamente para `FAILED`. Para obter mais informações, consulte Status de execução do trabalho do AWS Glue.	8 de novembro de 2023
AWS Glue Studio guia do usuário consolidado no AWS Glue guia do desenvolvedor	O AWS Glue Studio guia do usuário foi movido para o guia do desenvolvedor para criar um único guia de usuário unificado para AWS Glue Studio o AWS Glue console e o acesso AWS Glue Studio programático.	25 de outubro de 2023

Atualização da política AWSGlueServiceNotebookRole AWS gerenciada	Foram adicionadas informações sobre uma pequena atualização na política AWSGlueServiceNotebookRole AWS gerenciada. Para obter mais informações, consulte AWS GlueAtualizações nas políticas AWS gerenciadas.	9 de outubro de 2023
O AWS Glue Studio suporta cinco novas transformações integradas	O AWS Glue Studio suporta as cinco novas transformações integradas a seguir: Record matching, Remove null rows, Parse JSON column, Extract JSON path e Regex extractor. Para obter mais informações, consulte Edição de nós de transformação de dados AWS Glue gerenciados.	11 de agosto de 2023
Atualização da política AWSGlueServiceRole AWS gerenciada	Foram adicionadas informações sobre uma pequena atualização na política AWSGlueServiceRole AWS gerenciada. Para obter mais informações, consulte AWS GlueAtualizações nas políticas AWS gerenciadas.	4 de agosto de 2023

Compatibilidade com crawling em tabelas do Apache Hudi	Foram adicionadas informações sobre AWS Glue como usar para rastrear tabelas Hudi em buckets do Amazon S3 e registrar as tabelas Hudi no. AWS Glue Data Catalog Para obter mais informações, consulte Em quais armazenamentos de dados posso fazer crawling? e Propriedades do crawler.	21 de julho de 2023
Atualização da política AWSGlueConsoleFullAccess AWS gerenciada	Foram adicionadas informações sobre uma pequena atualização na política AWSGlueConsoleFullAccess AWS gerenciada. Para obter mais informações, consulte AWS GlueAtualizações nas políticas AWS gerenciadas.	14 de julho de 2023
Compatibilidade com crawling em tabelas do Apache Iceberg	Foram adicionadas informações sobre como usar AWS Glue para rastrear tabelas Iceberg em buckets do Amazon S3 e registrar as tabelas Iceberg no. AWS Glue Data Catalog Para obter mais informações, consulte Em quais armazenamentos de dados posso fazer crawling? e Propriedades do crawler.	7 de julho de 2023

Support for AWS Glue with Ray	Foram adicionadas informaçõ es sobre AWS Glue with Ray, um novo motor que pode apoiar AWS Glue trabalhos . Reorganizou o conteúdo existente AWS Glue com o Spark para eliminar a ambiguidade.	30 de maio de 2023
Support for AWS Glue Data Quality (GA)	AWS Glue A qualidade dos dados agora está disponíve l ao público em geral. AWS Glue A Qualidade dos Dados ajuda você a avaliar e monitorar a qualidade dos seus dados. Para obter informações sobre como usar a qualidade AWS Glue de dados com o catálogo de dados, consulte Qualidade de AWS Glue dados. Para saber mais sobre a qualidade AWS Glue dos dados para AWS Glue Studio, consulte Avaliação da qualidade dos dados com AWS Glue Studio.	24 de maio de 2023

Compatibilidade com tipos de operadores maiores para trabalhos do Apache Spark	Agora o uso dos tipos de operador `G.4X` e `G.8X` para trabalhos do Apache Spark é compatível. Esses tipos de operadores são adequados para trabalhos que têm workloads que contém as transformações, agregações, uniões e consultas mais exigentes. Para obter mais informações, consulte Adição de trabalhos no AWS Glue.	8 de maio de 2023
Compatibilidade com a criação de índices de partição ao fazer crawling em tabelas	Adicionadas informações sobre como os crawlers são compatíveis com a criação de índices de partição para as tabelas que o crawler detecta. Para obter mais informações, consulte Definir a opção de configuração do crawler de índice de partição.	24 de abril de 2023
Compatibilidade com métricas de uso de recursos	Foram adicionadas informações sobre a visualização do uso de recursos do serviço e a configuração de alarmes na Amazon. CloudWatch Para obter mais informações, consulte Monitorar recursos do AWS Glue.	7 de abril de 2023

Atualização da política AWSGlueConsoleFullAccess AWS gerenciada	Foram adicionadas informaçõ es sobre uma pequena atualização na política AWSGlueConsoleFullAccess AWS gerenciada. Para obter mais informações, consulte AWS GlueAtualizações nas políticas AWS gerenciadas.	28 de março de 2023
Orientação adicional para uso AWS Glue com um AWS SDK com exemplos	O Guia do AWS Glue desenvolvedor tem duas novas seções que fornecem informações para ajudá-lo a usar AWS Glue com um AWS SDK. Para obter mais informações, consulte Como usar AWS Glue com um AWS SDK e exemplos de código para AWS Glue usar AWS SDKs.	23 de fevereiro de 2023
Atualize a documentação do IAM com AWS Glue	Informações reorganizadas e adicionadas sobre o uso do IAM com AWS Glue. Para obter mais informações, consulte Gerenciamento de identidade e acesso do AWS Glue.	15 de fevereiro de 2023

Compatibilidade com execução de trabalhos de ETL de streaming no AWS Glue versão 4.0	Foram adicionadas informações sobre compatibilidade com execução de trabalhos de ETL de streaming no Glue versão 4.0 e novas opções para conexão com um cluster do Kafka ou um cluster do Amazon Managed Streaming for Apache Kafka e Amazon Kinesis Data Streams. Para obter mais informações, consulte Adicionar trabalhos de ETL de streaming no AWS Glue e Tipos de conexão e opções de ETL no AWS Glue.	8 de fevereiro de 2023
Compatibilidade com crawling em fontes de dados do MongoDB Atlas	Foram adicionadas informações sobre o uso AWS Glue para rastrear fontes de dados do MongoDB Atlas. Para obter mais informações, consulte Em quais armazenamentos de dados posso fazer crawling? , Propriedades de conexão do MongoDB e do MongoDB Atlas e Usar uma conexão do MongoDB ou do MongoDB Atlas.	6 de fevereiro de 2023

Compatibilidade com crawling de tabelas do Delta Lake usando um conector do Delta Lake	Foram adicionadas informações sobre como AWS Glue rastrear tabelas do Delta Lake usando um conector nativo do Delta Lake. Esse recurso permite que você use mecanismos de AWS consulta para consultar diretamente o registro de transações da Delta e usar recursos como viagem no tempo e garantias ACID, além de sincronizar seus metadados do Delta Lake dos arquivos de transações do Amazon S3 no catálogo de dados para permitir permissões de coluna em suas consultas no Lake Formation. Para obter mais informações, consulte Como especificar opções de configuração para um datastore Delta Lake e Consulta a tabelas Data Lake.	15 de dezembro de 2022

Support for AWS Glue Data Quality (versão prévia)	Support agora está disponível para AWS Glue Data Quality (versão prévia). AWS Glue A Qualidade de Dados ajuda você a avaliar e monitorar a qualidade dos seus dados quando você usa o AWS Glue 3.0. Para obter informações sobre como usar a Qualidade de AWS Glue Dados com o Catálogo de Dados, consulte Qualidade de AWS Glue dados (versão prévia). Para saber mais sobre a qualidade AWS Glue dos dados para AWS Glue Studio, consulte Avaliação da qualidade dos dados com AWS Glue Studio.	30 de novembro de 2022
Compatibilidade com um novo conector do Amazon Redshift Spark com novos recursos e melhorias de performance	Agora é possível usar um novo conector do Spark do Amazon Redshift com um novo driver JDBC com tarefas do AWS Glue ETL para criar aplicações do Apache Spark que leem e gravam dados no Amazon Redshift como parte de seus pipelines de ingestão e transformação de dados. Para obter mais informações, consulte Mover dados de e para o Amazon Redshift.	29 de novembro de 2022

Compatibilidade com o AWS Glue versão 4.0.	Informações adicionadas sobre compatibilidade com o AWS Glue versão 4.0. Os recursos incluem suporte nativo para estruturas de data lake aberto com Apache Hudi, Delta Lake e Apache Iceberg, e suporte nativo para o Cloud Shuffle Storage Plugin baseado no Amazon S3 (um plug-in do Apache Spark) para usar o Amazon S3 para capacidade de armazenam ento aleatório e elástico. Para obter mais informaçõ es, consulte as Notas de versão do AWS Glue e Migrar trabalho do AWS Glue para o AWS Glue versão 4.0.	28 de novembro de 2022
O AWS Glue Studio agora oferece transformações visuais personalizadas	As transformações visuais personalizadas permitem que os clientes definam, reutilizem e compartilhem a lógica de ETL específica da empresa entre suas equipes. Para obter mais informaçõ es, consulte Transformações visuais personalizadas.	28 de novembro de 2022

Compatibilidade com o uso do crawler do AWS Glue para publicar metadados para armazenamentos de dados JDBC	Agora é possível usar o crawler do AWS Glue para publicar metadados, como comentários e tipos brutos, no Data Catalog para armazenamentos de dados JDBC. Para obter mais informações, consulte Parâmetros definidos nas tabelas do Catálogo de Dados por rastreador, propriedades e estrutura do Crawler. JdbcTarget	18 de novembro de 2022
Compatibilidade com crawling de armazenamentos de dados do Snowflake	Agora, é possível usar o AWS Glue para rastrear de tabelas e visualizações do Snowflake e para publicar os metadados no Data Catalog como uma entrada de tabela. Para tabelas externas do Snowflake no Amazon S3, o crawler também rastreia locais do Amazon S3 e o tipo de formato de arquivo da tabela externa e os insere como parâmetros da tabela. Para obter mais informações, consulte Em quais armazenamentos de dados posso fazer crawling?, Propriedades de conexão do AWS Glue e Parâmetros definidos nas tabelas do Data Catalog pelo crawler.	18 de novembro de 2022

Compatibilidade com um melhor gerenciamento de ordem aleatória das aplicações do Spark	Agora é possível usar um novo Cloud Shuffle Storage Plugin for Apache Spark. Para obter mais informações, consulte Gerenciador de ordem aleatória do Spark no AWS Glue com o Amazon S3 e Cloud Shuffle Storage Plugin for Apache Spark.	15 de novembro de 2022
Adicionada compatibilidade com destinos do Data Catalog ao acelerar notificações de eventos do Amazon S3 de crawls	Além da compatibilidade já existente com destinos do Amazon S3, agora é possível acelerar crawls para destinos do Data Catalog usando notificações de eventos do Amazon S3. Para obter mais informações, consulte Aceleração de crawls usando notificações de eventos do Amazon S3.	13 de outubro de 2022
Compatibilidade com a especificação do número máximo de tabelas que um crawler pode criar	Agora, há compatibilidade com a especificação do número máximo de tabelas que o crawler tem permissão para criar. Para obter mais informações, consulte Como especificar o número máximo de tabelas que o crawler pode criar.	6 de setembro de 2022

Compatibilidade com Python 3.9 em trabalhos de shell do Python no AWS Glue	Agora, o AWS Glue oferece suporte para a execução de scripts compatíveis com o Python 3.9 em trabalhos de shell do Python e para optar pelo uso de conjuntos pré-empacotados de bibliotecas. Para obter mais informações, consulte Trabalhos de shell do Python no AWS Glue.	11 de agosto de 2022
Support para execução de AWS Glue trabalhos não urgentes ou não urgentes em termos de tempo em capacidade ociosa	Agora, há suporte para a configuração de execuções flexíveis de trabalho para trabalhos não urgentes, como trabalhos de pré-produção, testes e cargas de dados ocasionais. Para obter mais informações, consulte Adição de trabalhos no AWS Glue.	9 de agosto de 2022
Suporte para um novo tipo de operador para trabalhos de streaming	O suporte agora está disponível para uso do tipo de operador `G.025X` para trabalhos de streaming de baixo volume. Para obter mais informações, consulte Adição de trabalhos no AWS Glue.	14 de julho de 2022
Suporte para o uso do Kafka SASL em conexões do AWS Glue	O suporte agora está disponível para uso do Kafka SASL em conexões doAWS Glue. Para obter mais informações, consulte Propriedades de conexão do AWS Glue Kafka para autenticação do cliente.	5 de julho de 2022

Compatibilidade com o conector do Apache Kafka para esquemas protobuf	O suporte para o Apache Kafka Connector agora está disponível para esquemas Protobuf. Para obter mais informações, consulte Registro de esquemas do AWS Glue.	9 de junho de 2022
Compatibilidade com Auto Scaling para trabalhos do AWS Glue (disponível ao público)	Adição de informações sobre como usar o Auto Scaling para trabalhos no AWS Glue versão 3.0 para escalar dinamicamente os recursos de computação. Para mais informações, consulte Uso de Auto Scaling para o AWS Glue.	14 de abril de 2022
Atualização na documentação para desenvolvimento e testes do AWS Glue de scripts de trabalho doAWS Glue	Informações reorganizadas e adicionadas sobre os métodos de desenvolvimento e testes disponíveis para o AWS Glue, incluindo instruções de desenvolvimento com o Docker. Para mais informaçõ es, consulte Desenvolvimento e teste de scripts de trabalho do AWS Glue.	14 de março de 2022
Adição de protocol buffers (protobuf) como formato de dados compatível para o registro de esquemas do AWS Glue	Adicionadas informaçõ es sobre o Protobuf como formato de dados suportado (além do AVRO e JSON). Para obter mais informaçõ es, consulte Registro de esquemas do AWS Glue.	25 de fevereiro de 2022

Suporte ao crawling de tabelas do Delta Lake	Foram adicionadas informaçõ es sobre como usar AWS Glue para rastrear as tabelas do Delta Lake. Para obter mais informações, consulte Como especificar opções de configuração para um datastore do Delta Lake.	24 de fevereiro de 2022
Support para insights AWS Glue de emprego	Foram adicionadas informaçõ es sobre o uso AWS Glue de insights de tarefas para simplificar a depuração e a otimização de tarefas. AWS Glue Para obter mais informações, consulte Monitorar com insights de trabalhos do AWS Glue.	8 de fevereiro de 2022
Suporte a crawling de tabelas de catálogo de dados baseadas no Amazon S3 usando um endpoint da VPC	Além de armazenamentos de dados do Amazon S3, você pode configurar suas tabelas de catálogo de dados baseadas no Amazon S3 para serem acessadas somente por um ambiente da Amazon Virtual Private Cloud (Amazon VPC) para fins de segurança , auditoria ou controle. Para obter mais informações, consulte Crawling de um datastore do Amazon S3 ou de tabelas de catálogo de dados baseado no Amazon S3 usando um endpoint da VPC.	3 de fevereiro de 2022

Suporte a tabelas governadas pelo Lake Formation	Adição de informações sobre o suporte do AWS Glue para tabelas governadas pelo Lake Formation, com suporte a transações ACID, compactação automática de dados e consultas de viagem no tempo. Para obter mais informações, consulte a API do AWS Glue e o Guia do desenvolvedor do AWS Lake Formation.	30 de novembro de 2021
Novas políticas AWS gerenciadas adicionadas para sessões interativas e notebooks	As novas políticas gerenciadas para IAM forneceram segurança aprimorada para uso AWS Glue com sessões interativas e notebooks. Para obter mais informações, consulte Políticas gerenciadas pela AWS para o AWS Glue.	30 de novembro de 2021
O registro de esquemas do Glue agora oferece suporte a trabalhos de transmissão	Você pode criar trabalhos de transmissão que acessem tabelas que fazem parte do Glue Schema Registry. Para obter mais informações, consulte AWS Glue Schema Registry e Adição de trabalhos de ETL de transmissão no AWS Glue.	15 de novembro de 2021

Suporte a novos recursos de machine learning	Adicionadas informações sobre novos recursos para a transformação de machine learning de correspondências de Find (Localizar), incluindo correspondência incremental e pontuação de correspondência. Para obter mais informações, consulte Localização de correspondências incrementais e Estimação da qualidade das correspondências usando as pontuações de confiança de correspondência.	31 de outubro de 2021
(prévia privada) Suporte para trabalhos flexíveis do AWS Glue	Adição de informações sobre a configuração de trabalhos do Spark no AWS Glue com uma classe de execução flexível, adequada para trabalhos insensíveis ao tempo, cujos horários de início e conclusão podem variar. Para obter mais informações, consulte Adição de trabalhos no AWS Glue.	29 de outubro de 2021
Compatibilidade com a aceleração de crawls usando notificações de eventos do Amazon S3	Adição de informações sobre a aceleração de crawls usando notificações de eventos do Amazon S3. Para obter mais informações, consulte Aceleração de crawls usando notificações de eventos do Amazon S3.	15 de outubro de 2021

Opções de configuração de segurança adicionais relacionadas ao controle de acesso e a VPCs	Adição de informações sobre como você pode configurar novas permissões de controle de acesso no AWS Glue e configurar VPCs. Para obter mais informações, consulte AWSTags em AWS Glue, Políticas baseadas em identidade (políticas do IAM) que controlam as configurações usando chaves de condição ou chaves de contexto e Configuração de todas as AWS chamadas para passar pela sua VPC.	13 de outubro de 2021
Suporte para políticas de endpoint da VPC	Adição de informações sobre o suporte a políticas de endpoint da Virtual Private Cloud (VPC) no AWS Glue. Para obter mais informações, consulte Endpoints de VPC do AWS Glue e interface (AWS PrivateLink).	11 de outubro de 2021
O Glue Studio agora está disponível na China	O AWS Glue Studio agora está disponível nas regiões Pequim e Ningxia da China.	11 de outubro de 2021

O AWS Glue Studio oferece criação de cadernos, para edição interativa de trabalhos	Os cadernos ajudam você a escrever e executar código, visualizar os resultados e compartilhar ideias. Normalmente, os cientistas de dados usam cadernos para experimentos e tarefas de exploração de dados. Para obter mais informações, consulte Usar cadernos.	1.º de outubro de 2021
O acesso direto a fontes de transmissão agora está disponível	Ao adicionar origens de dados ao seu trabalho de ETL no editor visual, você pode fornecer informações para acessar o fluxo de dados em vez de usar um banco de dados e uma tabela do Data Catalog.	30 de setembro de 2021
Documentada a política de suporte à versão do AWS Glue	Adicionadas informações sobre a política de suporte à versão do AWS Glue e as fases de fim de vida para certas versões do AWS Glue. Para obter mais informações, consulte a Política de suporte à versão do AWS Glue.	24 de setembro de 2021
Conectores personalizados agora podem ser usados com visualizações de dados	Ao editar o nó da origem dos dados usando um conector personalizado, você pode visualizar o conjunto de dados escolhendo a guia de visualização de dados. Para obter mais informações, consulte Conectores personalizados .	24 de setembro de 2021

Support para sessões AWS Glue interativas (versão prévia privada)	(Pré-visualização privada) Foram adicionadas informações sobre o uso de sessões AWS Glue interativas para executar cargas de trabalho do Spark na nuvem a partir de qualquer notebook Jupyter. As sessões interativas são o método preferido para desenvolver seu código de AWS Glue extração, transformação e carregamento (ETL) quando você usa AWS Glue 2.0 ou posterior. Para obter mais informações, consulte Configurando e executando sessões AWS Glue interativas para o Jupyter Notebook.	24 de agosto de 2021
Compatibilidade com a criação de fluxos de trabalho com base em esquemas (disponível ao público)	Adição de informações sobre a codificação de casos de uso comuns de extração, transformação e carregamento (ETL) nos blueprints e, em seguida, a criação de fluxos de trabalho de blueprints. Permite que os analistas de dados criem e executem facilmente processos de ETL complexos. Para obter mais informações, consulte Realização de atividades de ETL complexas usando esquemas e fluxos de trabalho no AWS Glue.	23 de agosto de 2021

Compatibilidade com o AWS Glue versão 3.0.	Adição de informações sobre o suporte para o AWS Glue versão 3.0, que é compatível com a atualização do mecanismo Apache Spark 3.0 para executar trabalhos de ETL do Apache Spark, além de outras otimizações e atualizações. Para obter mais informações, consulte Notas de release do AWS Glue e Migrar trabalho do AWS Glue para o AWS Glue versão 3.0. Outros recursos nesta versão incluem o gerenciador de ordem aleatória do AWS Glue, um leitor CSV vetorizado SIMD e predicados de partição de catálogo. Para obter mais informações, consulte o Gerenciador de ordem aleatória do Spark do AWS Glue com o Amazon S3, Opções de formato para entradas e saídas de ETL no AWS Glue e Filtragem do lado do servidor usando predicados de partição de catálogo.	18 de agosto de 2021
AWS GovCloud (US) Region	AWS Glue Studioagora está disponível no AWS GovCloud (US) Region	18 de agosto de 2021

Criação de shell do Python disponível no AWS Glue Studio	Agora, ao criar um novo trabalho, você pode optar por criar um trabalho do shell do Python. Para obter mais informações, consulte Iniciar o processo de criação de trabalho e Editar trabalhos de shell do Python no AWS Glue Studio.	13 de agosto de 2021
Support para iniciar um fluxo de trabalho com um EventBridge evento da Amazon	Adicionadas informações sobre como o AWS Glue pode ser um consumidor de eventos em uma arquitetura orientada a eventos. Para obter mais informações, consulte Iniciando um AWS Glue fluxo de trabalho com um EventBridge evento da Amazon e Visualizando os EventBridge eventos que iniciaram um fluxo de trabalho.	14 de julho de 2021
Adição do JSON como um formato de dados com suporte para o AWS Glue Schema Registry	Adicionadas informações sobre o JSON como um formato de dados suportado (além do AVRO). Para obter mais informações, consulte Registro de esquemas do AWS Glue.	30 de junho de 2021

Criar trabalhos de transmissão do AWS Glue sem uma tabela do Data Catalog	A função `create_data_frame_from_options` do Python ou `getSource` para scripts Scala é compatível com a criação de trabalhos de ETL de transmissão que fazem referência aos fluxos de dados diretamente, em vez de exigir uma tabela do Data Catalog.	15 de junho de 2021
AWS Glueas transformações de aprendizado de máquina agora oferecem suporte AWS Key Management Service a chaves	Você pode especificar uma configuração ou AWS KMS chave de segurança ao configurar as transform ações do AWS Glue Machine Learning com o console, a CLI ou as APIs. AWS Glue Para obter mais informaçõ es, consulte Usar a criptogra fia dos dados com transform ações de machine learning e API de machine learning do AWS Glue.	15 de junho de 2021
Atualização da política AWSGlueConsoleFullAccess AWS gerenciada	Foram adicionadas informaçõ es sobre uma pequena atualização na política AWSGlueConsoleFullAccess AWS gerenciada. Para obter mais informações, consulte AWS GlueAtualizações nas políticas AWS gerenciadas.	10 de junho de 2021

Visualizar o conjunto de dados do trabalho ao criar e editar trabalhos	Você pode usar a nova guia Data preview (Previsualização de dados) para um nó no diagrama de trabalhos a fim de ver uma amostra dos dados processados por esse nó. Para obter mais informaçõ es, consulte Usar previsual izações de dados no editor de trabalhos visual.	7 de junho de 2021
Compatibilidade com a especificação de um valor que indica o local da tabela para a saída do crawler.	Informações adicionadas sobre a especificação de um valor que indica o local da tabela ao configurar a saída do crawler. Para obter mais informações, consulte Como especificar o local da tabela.	04 de junho de 2021
Compatibilidade com crawling em uma amostra de arquivos em um conjunto de dados ao realizar o crawling em um datastore do Amazon S3	Informações adicionad as sobre crawling de uma amostra de arquivos ao realizar o crawling no Amazon S3. Para obter mais informaçõ es, consulte Propriedades do crawler.	10 de maio de 2021

Suporte para o gravador de parquet otimizado do AWS Glue	Foram adicionadas informaçõ es sobre o uso do gravador de parquetes AWS Glue otimizado DynamicFrames para criar ou atualizar tabelas com a `parquet` classific ação. Para obter mais informações, consulte Criar tabelas, atualizar esquemas e adicionar novas partições no catálogo de dados de trabalhos de ETL do AWS Glue e Opções de formato para entradas e saídas de ETL no AWS Glue.	4 de maio de 2021
Compatibilidade com senhas de autenticação de cliente do Kafka	Informações adicionadas sobre trabalhos de ETL de transmissão no AWS Glue suportam autenticação de certificado de cliente SSL com produtores de transmissão do Apache Kafka. Agora você pode fornecer um certificado personalizado ao definir uma conexão do AWS Glue com um cluster do Apache Kafka, que o AWS Glue usará ao autenticar com ele. Para obter mais informações, consulte Propriedades de conexão do AWS Glue e API de conexão.	28 de abril de 2021

Compatibilidade com o consumo de dados do Amazon Kinesis Data Streams em outra conta em trabalhos de ETL de transmissão	Informações adicionadas sobre como criar um trabalho de ETL de transmissão para consumir dados do Amazon Kinesis Data Streams em outra conta. Para obter mais informações, consulte Adicionar trabalhos ETL de streaming no AWS Glue.	30 de março de 2021
Transformação SQL disponível	Você pode usar um nó de transformação SQL para gravar sua própria transformação na forma de uma consulta SQL. Para obter mais informações, consulte Usar uma consulta SQL para transformar dados.	23 de março de 2021
Suporte para a criação de fluxos de trabalho de esquemas (pré-visualização pública)	(Previsualização pública) Informações adicionadas sobre a codificação de casos de uso comuns de extração, transformação e carregamento (ETL) nos blueprints e subsequente criação de fluxos de trabalho de blueprints. Permite que os analistas de dados criem e executem facilmente processos de ETL complexos. Para obter mais informações, consulte Realização de atividades de ETL complexas usando esquemas e fluxos de trabalho no AWS Glue.	22 de março de 2021

Conectores podem ser usados para destinos de dados	Agora há suporte para o uso de um AWS Marketplace conector ou personalizado para seu destino de dados. Para obter mais informações, consulte Criação de trabalhos com conectores personalizados.	15 de março de 2021
Suporte para métricas de importância de coluna para transformações de machine learning do AWS Glue	Informações adicionadas sobre a exibição de métricas de importância de coluna ao trabalhar com transformações de machine learning do AWS Glue. Para obter mais informações, consulte Trabalhar com transformações de machine learning no console do AWS Glue	5 de fevereiro de 2021
O recurso de programação de trabalhos agora está disponível no AWS Glue Studio	Você pode definir uma programação baseada em tempo para suas execuções de trabalhos no AWS Glue Studio. Você pode usar o console para criar uma programação básica ou definir uma programação mais complexa usando a sintaxe cron. Para obter mais informações, consulte Programar execuções de trabalhos.	21 de dezembro de 2020

Lançamento de conectores personalizados do AWS Glue	Os conectores personalizados do AWS Glue permitem que você descubra e assine conectores no AWS Marketplace. Também lançamos interfaces do runtime do Spark no AWS Glue para ligar conectores criados para a fonte de dados do Apache Spark, consulta federada do Athena e APIs do JDBC. Para obter mais informações, consulte Uso de conectores e conexões com o AWS Glue Studio.	21 de dezembro de 2020
Suporte para execução de trabalhos de ETL de transmissão no AWS Glue versão 2.0	Informações adicionadas sobre suporte para a execução de trabalhos de ETL de transmissão no Glue versão 2.0. Para obter mais informações, consulte Adicionar trabalhos ETL de streaming no AWS Glue.	18 de dezembro de 2020

Compatibilidade com o particionamento de workload com execução limitada	Informações adicionadas sobre como habilitar o particionamento de workload para configurar os limites superiores no tamanho do conjunto de dados ou o número de arquivos processados em execuções de trabalho de ETL. Para obter mais informações, consulte Particionamento de workload com execução limitada.	23 de novembro de 2020
Compatibilidade com gerenciamento aprimorado de partições	Informações adicionadas sobre como usar novas APIs para adicionar ou excluir um índice de partição para/de uma tabela existente. Para obter mais informações, consulte Trabalhar com índices de partição.	23 de novembro de 2020
Suporte para o AWS Glue Schema Registry	Informações adicionadas sobre o uso do registro de esquema do AWS Glue para descobrir, controlar e evoluir esquemas centralmente. Para obter mais informações, consulte Registro de esquemas do AWS Glue.	19 de novembro de 2020

Compatibilidade com o formato de entrada Grok em trabalhos de ETL de transmissão	Informações adicionadas sobre a aplicação de padrões Grok a fontes de transmissão, como arquivos de log. Para obter mais informações, consulte Aplicar padrões Grok a fontes de transmissão.	17 de novembro de 2020
Suporte para a adição de tags a fluxos de trabalho no console do AWS Glue	Informações adicionadas sobre como adicionar tags ao criar um fluxo de trabalho usando o console do AWS Glue. Para obter mais informações, consulte Criar e desenvolver um fluxo de trabalho usando o console do AWS Glue.	27 de outubro de 2020
Compatibilidade com execuções incrementais de crawler	Informações adicionadas sobre o suporte para execuções incrementais de crawler, que rastreiam somente as pastas do Amazon S3 adicionadas desde a última execução. Para obter mais informações, consulte Crawls incrementais.	21 de outubro de 2020

Suporte para detecção de esquema para fontes de dados de ETL de transmissão. Suporte para transmissão Avro de fontes de dados de ETL e Kafka autogerenciado	A transmissão de trabalhos de extração, transformação e carregamento (ETL) no AWS Glue agora pode detectar automaticamente o esquema de registros de entrada e lidar com alterações de esquema por registro. Origens dos dados do Kafka autogerenciado agora são suportadas. Trabalhos de ETL de transmissão agora suportam o formato Avro nas origens dos dados. Para obter mais informações, consulte ETL de transmissão no AWS Glue, Definir propriedades de trabalho para um trabalho de ETL de transmissão e Notas e restrições para fontes de transmissão do Avro.	7 de outubro de 2020
Compatibilidade com crawling em origens de dados do MongoDB e DocumentDB	Informações adicionadas sobre suporte para crawling de origens dos dados do MongoDB e do Amazon DocumentDB (compatível com MongoDB). Para obter mais informações, consulte Definir crawlers.	5 de outubro de 2020

Compatibilidade com conformidade com FIPS	Informações adicionadas sobre endpoints dos FIPS para clientes que precisam de módulos criptográficos validados pelo FIPS 140-2 ao acessar dados usando o AWS Glue. Para obter mais informações, consulte Conformidade com os FIPS.	23 de setembro de 2020
O AWS Glue Studio fornece uma interface visual fácil de usar para criar e monitorar trabalhos	Agora você pode usar uma interface simples baseada em gráficos para compor trabalhos que movem e transformam dados e executá-los no AWS Glue. Em seguida, você pode usar o painel de execução de tarefas no AWS Glue Studio para monitorar a execução de ETL e garantir que seus trabalhos estejam operando conforme pretendido. Para obter mais informações, consulte o Guia do usuário do AWS Glue Studio.	23 de setembro de 2020
Compatibilidade com a criação de índices de tabela para melhorar a performance de consulta	Informações adicionadas sobre a criação de índices de tabela para permitir que você recupere um subconjunto das partições de uma tabela. Para obter mais informações, consulte Trabalhar com índices de partição.	9 de setembro de 2020

Compatibilidade com tempos reduzidos de inicialização ao executar trabalhos de ETL do Apache Spark no AWS Glue versão 2.0.	Informações adicionadas sobre o suporte para o AWS Glue versão 2.0 que fornece uma infraestrutura atualizada para executar trabalhos do ETL do Apache Spark com tempos de inicialização reduzidos, alterações no registro e suporte para especificar módulos Python adicionais no nível do trabalho. Para obter mais informações, consulte Notas de release do AWS Glue e Executar trabalhos de ETL do Spark com startup reduzidos.	10 de agosto de 2020
Compatibilidade com a limitação do número de execuções simultâneas de fluxo de trabalho.	Informações adicionadas sobre como limitar o número de execuções de fluxo de trabalho simultâneas para um determinado fluxo de trabalho. Para obter mais informações, consulte Criar e desenvolver um fluxo de trabalho usando o console do AWS Glue.	10 de agosto de 2020

Compatibilidade com crawling em um datastore do Amazon S3 usando um endpoint da VPC	Informações adicionadas sobre a configuração do datastore do Amazon S3 para acesso somente por um ambiente da Amazon Virtual Private Cloud (Amazon VPC), para fins de segurança , auditoria ou controle. Para obter mais informações, consulte Crawling de um datastore do Amazon S3 usando um endpoint da VPC.	7 de agosto de 2020
Compatibilidade com a retomada de execuções de fluxo de trabalho	Informações adicionad as sobre como retomar execuções de fluxo de trabalho que foram concluída s apenas parcialmente porque um ou mais nós (trabalhos ou crawlers) não foram concluído s com êxito. Para obter mais informações, consulte Reparar e retomar uma execução de fluxo de trabalho.	27 de julho de 2020
Compatibilidade com a habilitação de certificados de CA privada em conexões do Kafka no AWS Glue.	Informações adicionadas sobre novas opções de conexão que são compatíve is com a habilitação de certificados CA privados para conexões Kafka no AWS Glue. Para obter mais informações, consulte Tipos de conexão e opções para ETL no AWS Glue e Parâmetros especiais usados pelo AWS Glue.	20 de julho de 2020

Compatibilidade com a leitura de dados do DynamoDB em outra conta	Informações adicionadas sobre o suporte do AWS Glue para leitura de dados de tabela do DynamoDB de outra conta da AWS . Para obter mais informações, consulte Leitura de dados do DynamoDB em outra conta.	17 de julho de 2020
Compatibilidade com uma conexão de gravador do DynamoDB no AWS Glue versão 1.0 ou posterior	Informações adicionadas sobre o suporte ao gravador do DynamoDB e opções de conexão novas ou atualizadas para o DynamoDB ler ou gravar. Para obter mais informações, consulte Tipos de conexão e opções para ETL no AWS Glue.	17 de julho de 2020
Compatibilidade com links de recursos e controle de acesso entre contas usando o AWS Glue e o Lake Formation	Conteúdo adicionado sobre novos objetos do Data Catalog chamados links de recursos, e sobre como gerenciar o compartilhamento de recursos do Data Catalog entre contas com o AWS Glue e o AWS Lake Formation. Para obter mais informações, consulte Concessão de acesso entre contas e Links de recursos de tabela.	7 de julho de 2020

Compatibilidade com amostragem de registros ao realizar crawling dos armazenamentos de dados do DynamoDB	Adicionadas informações sobre novas propriedades que você pode configurar ao realizar crawling de um datastore do DynamoDB. Para obter mais informaçõ es, consulte Propriedades do crawler.	12 de junho de 2020
Compatibilidade com a interrupção de uma execução de fluxo de trabalho.	Informações adicionadas sobre como interromper a execução de determinado fluxo de trabalho. Para obter mais informações, consulte Interromper uma execução de fluxo de trabalho.	14 de maio de 2020
Compatibilidade com trabalhos de ETL de transmissão do Spark	Inclusão de informações sobre a criação de trabalhos de extração, transformação e carregamento (ETL) com fontes de dados de streaming . Para obter mais informações, consulte Adicionar trabalhos ETL de streaming no AWS Glue.	27 de abril de 2020

Compatibilidade com a criação de tabelas, atualização do esquema e adição de novas partições ao Data Catalog após a execução de um trabalho de ETL	Adição de informações sobre como você pode habilitar a criação de tabelas, atualizar o esquema e adicionar novas partições para visualizar os resultados do trabalho de ETL no catálogo de dados. Para obter mais informações, consulte Criar tabelas, atualizar esquemas e adicionar novas partições no Data Catalog de trabalhos de ETL do AWS Glue.	2 de abril de 2020
Compatibilidade com a especificação de uma versão para o formato de dados Apache Avro como uma entrada e saída de ETL no AWS Glue	Adição de informações sobre como especificar uma versão para o formato de dados do Apache Avro como uma entrada e saída de ETL no AWS Glue. A versão padrão é 1.7. É possível usar a opção de formato `version` para especificar o Avro versão 1.8 para habilitar a leitura/gravação lógica. Para obter mais informações, consulte Opções de formato para entradas e saídas de ETL no AWS Glue.	31 de março de 2020

Compatibilidade com o committer otimizado para EMRFS do S3 para gravação de dados Parquet no Amazon S3	Informações adicionadas sobre como definir um novo sinalizador para habilitar o committer otimizado para EMRFR S3, para gravar dados parquet no Amazon S3 ao criar ou atualizar um trabalho do AWS Glue. Para obter mais informações, consulte Parâmetros especiais usados pelo AWS Glue.	30 de março de 2020
Support for machine learning se transforma em um recurso gerenciado por tags AWS de recursos	Foram adicionadas informaçõ es sobre o uso AWS de tags de recursos para gerenciar e controlar o acesso às suas transformações de aprendiza do de máquina emAWS Glue. Você pode atribuir tags AWS de recursos a trabalhos, gatilhos, endpoints, rastreado res e transformações de aprendizado de máquina em. AWS Glue Para obter mais informações, consulte Tags da AWS no AWS Glue.	2 de março de 2020
Compatibilidade com argumentos de trabalho não substituíveis	Adição de informações sobre suporte para parâmetros especiais de trabalho que não podem ser substituídos em triggers ou quando o trabalho é executado. Para obter mais informações, consulte Adição de trabalhos no AWS Glue.	12 de fevereiro de 2020

Compatibilidade com novas transformações para conjuntos de dados no Amazon S3	Informações adicionadas sobre novas transformações (Merge, Purge e Transitio n) e exclusões de classe de armazenamento do Amazon S3 para que aplicações do Apache Spark funcionem com conjuntos de dados no Amazon S3. Para obter mais informações sobre o suporte a essas transformações para Python, mergeDyna micFrameconsulte Como trabalhar com conjuntos de dados no Amazon S3. Para Scala, consulte mergeDyna micFramese as APIs do AWS GlueScala GlueContext .	16 de janeiro de 2020
Compatibilidade com a atualização do Data Catalog com novas informações de partição de um trabalho de ETL	Foram adicionadas informaçõ es sobre como codificar um script de extração, transform ação e carregamento (ETL) para atualizá-lo AWS Glue Data Catalog com novas informações de partição. Com esse recurso, não é mais necessário executar novamente o crawler após a conclusão do trabalho para exibir as novas partições. Para obter mais informaçõ es, consulte Atualizar o Data Catalog com novas partições.	15 de janeiro de 2020

Novo tutorial: Usando um SageMaker notebook	Foi adicionado um tutorial que demonstra como usar um SageMaker notebook da Amazon para ajudar a desenvolver seus scripts de ETL e aprendizado de máquina. Consulte o tutorial: Use um SageMaker notebook da Amazon com seu endpoint de desenvolvimento.	3 de janeiro de 2020
Compatibilidade com leitura do MongoDB e do Amazon DocumentDB (compatível com MongoDB)	Informações adicionadas sobre novos tipos e opções de conexão para ações de leitura e gravação no MongoDB e no Amazon DocumentDB (compatível com MongoDB). Para obter mais informações, consulte Tipos de conexão e opções para ETL no AWS Glue.	17 de dezembro de 2019

Várias correções e esclarecimentos	Adição de correções e esclarecimentos ao longo do documento. Itens removidos do capítulo Problemas conhecidos. Adição de avisos informando que o AWS Glue oferece suporte apenas a chaves mestras do cliente (CMKs) simétricas ao especificar configurações de criptografia do Catálogo de Dados e criar configurações de segurança. Adição de uma nota informando que o AWS Glue não permite gravar no Amazon DynamoDB.	9 de dezembro de 2019
Compatibilidade com drivers JDBC personalizados	Foram adicionadas informações sobre como conectar destinos e fontes de dados com drivers JDBC que não são compatíveis nativamente com o AWS Glue, como MySQL versão 8 e Oracle Database versão 18. Para obter mais informações, consulte Valores de connectionType do JDBC.	25 de novembro de 2019

Support para conectar SageMaker notebooks a diferentes endpoints de desenvolvimento	Foram adicionadas informações sobre como você pode conectar um SageMaker notebook a diferentes endpoints de desenvolvimento. Atualizações para descrever a nova ação do console para mudar para um novo endpoint de desenvolvimento e a nova política SageMaker do IAM. Para obter mais informações, consulte Trabalhando com notebooks no AWS Glue console e Criar uma política do IAM para Amazon SageMaker Notebooks.	21 de novembro de 2019
Compatibilidade com a versão do AWS Glue em transformações de machine learning	Informações adicionadas sobre como definir a versão do AWS Glue em uma transformação de machine learning de maneira a indicar com qual versão do AWS Glue uma transformação de machine learning é compatível. Para obter mais informações, consulte Trabalhar com transformações de machine learning no console do AWS Glue.	21 de novembro de 2019

Compatibilidade com a recuperação de estados anteriores de seus marcadores de trabalho	Adição de informações sobre como retroceder seus marcadores de trabalho para qualquer execução de trabalho anterior, resultando no reprocessamento subsequente de dados de execução de trabalho somente a partir da execução do trabalho marcado. Descrição de duas novas subopções para a opção `job-bookmark-pause` que permitem executar um trabalho entre dois marcadores. Para obter mais informações, consulte Rastrear dados processados usando marcadores de trabalho e Parâmetros especiais usados pelo AWS Glue.	22 de outubro de 2019
Compatibilidade com certificados JDBC personalizados para conexão com um datastore	Adição de informações sobre o suporte do AWS Glue a certificados JDBC personalizados para conexões SSL a fontes de dados ou destinos do AWS Glue. Para obter mais informações, consulte Trabalhar com conexões no console do AWS Glue.	10 de outubro de 2019

Compatibilidade com o formato de distribuição wheel do Python	Informações adicionadas sobre o suporte do AWS Glue a arquivos wheel (junto com arquivos egg) como dependências para trabalhos de shell do Python. Para obter mais informações, consulte Fornecer sua própria bibliotec a Python.	26 de setembro de 2019
Compatibilidade com versionamento de endpoints de desenvolvimento no AWS Glue	Adição de informações sobre como definir o `Glue version` em endpoints de desenvolvimento. O `Glue version` determina as versões de Apache Spark e Python que são compatíveis com o AWS Glue. Para obter mais informações, consulte Adicionar um endpoint de desenvolvimento.	19 de setembro de 2019
Compatibilidade com monitoramento do AWS Glue usando a interface de usuário do Spark	Adição de informações sobre como usar a IU do Apache Spark para monitorar e depurar trabalhos de ETL do AWS Glue em execução no sistema de trabalhos do AWS Glue e aplicativos Spark em endpoints de desenvolvimento do AWS Glue. Para obter mais informações, consulte Monitorar o AWS Glue usando a interface do usuário do Spark.	19 de setembro de 2019

Aprimoramento do suporte para desenvolvimento de script de ETL local usando a biblioteca de ETL pública do AWS Glue	Atualização do conteúdo da biblioteca de ETL do AWS Glue para refletir que o AWS Glue versão 1.0 agora é compatível. Para obter mais informações, consulte Desenvolver e testar scripts de ETL localmente usando a biblioteca de ETL do AWS Glue.	18 de setembro de 2019
Compatibilidade com a exclusão de classes de armazenamento do Amazon S3 ao executar trabalhos	Informações adicionadas sobre a exclusão de classes de armazenamento do Amazon S3 ao executar trabalhos de ETL do AWS Glue que leem arquivos ou partições do Amazon S3. Para obter mais informações, consulte Excluir classes de armazenamento do Amazon S3.	29 de agosto de 2019
Suporte para desenvolvimento de script de ETL local usando a biblioteca de ETL pública do AWS Glue	Adição de informações sobre como desenvolver e testar scripts Python e Scala de ETL localmente sem a necessida de de uma conexão de rede. Para obter mais informaçõ es, consulte Desenvolver e testar scripts de ETL localment e usando a biblioteca de ETL do AWS Glue.	28 de agosto de 2019

Problemas conhecidos	Adição de informações sobre problemas conhecidos no AWS Glue. Para obter mais informações, consulte Problemas conhecidos do AWS Glue,	28 de agosto de 2019
Compatibilidade com transformações de machine learning no AWS Glue	Adição de informações sobre recursos de machine learning fornecidos pelo AWS Glue para criar transformações personalizadas. É possível criar essas transformações ao criar uma tarefa. Para obter mais informações, consulte Transformações de machine learning no AWS Glue.	8 de agosto de 2019
Compatibilidade com Amazon Virtual Private Cloud compartilhada	Informações adicionadas sobre o suporte do AWS Glue para Amazon Virtual Private Cloud compartilhada. Para obter mais informações, consulte Amazon VPCs compartilhadas.	6 de agosto de 2019
Compatibilidade com versionamento no AWS Glue	Informações adicionadas sobre como definir a `Glue version` em endpoints de desenvolvimento. A versão do AWS Glue determina as versões de Apache Spark e Python que são compatíveis com o AWS Glue. Para obter mais informações, consulte Adição de trabalhos no AWS Glue.	24 de julho de 2019

Compatibilidade com opções adicionais de configuração para endpoints de desenvolvimento	Adição de informações sobre opções de configuração para endpoints de desenvolvimento que têm cargas de trabalho com uso intensivo de memória. Você pode escolher entre duas novas configurações que fornecem mais memória por executor. Para obter mais informações, consulte Trabalhar com endpoints de desenvolvimento no console do AWS Glue.	24 de julho de 2019
Compatibilidade com atividades de extração, transferência e carregamento (ETL) usando fluxos de trabalho	Informações adicionadas sobre como usar uma nova construção chamada de fluxo de trabalho para criar uma atividade complexa de vários trabalhos de extração, transformação e carregamento (ETL) que o AWS Glue pode executar e rastrear como uma entidade única. Para obter mais informações, consulte Realizar atividades complexas de ETL usando fluxos de trabalho no AWS Glue.	20 de junho de 2019

Compatibilidade com Python 3.6 em tarefas de shell do Python	Adicionadas informações sobre o suporte a Python 3.6 em tarefas de shell do Python. É possível especificar Python 2.7 ou Python 3.6 como uma propriedade da tarefa. Para obter mais informações, consulte Adicionar trabalhos de shell do Python no AWS Glue.	5 de junho de 2019
Compatibilidade com endpoints de nuvem privada virtual (VPC)	Informações adicionadas sobre como se conectar diretamente ao AWS Glue por meio de um endpoint de interface na VPC. Quando você usa um endpoint de interface de VPC, a comunicação entre a VPC e o AWS Glue é realizada integralmente e com segurança na rede da AWS . Para obter mais informações, consulte Usar o AWS Glue com endpoints da VPC.	4 de junho de 2019

Suporte para o registro de logs contínuo e em tempo real para trabalhos do AWS Glue	Foram adicionadas informaçõ es sobre como ativar e visualizar os registros de tarefas do Apache Spark em tempo real, CloudWatc h incluindo os registros do driver, cada um dos registros do executor e uma barra de progresso do trabalho do Spark. Para obter mais informações, consulte Registro contínuo para tarefas do AWS Glue.	28 de maio de 2019
Compatibilidade com tabelas existentes do Data Catalog como fontes do crawler	Informações adicionadas sobre como especificar uma lista de tabelas existente s do Data Catalog como fontes do crawler. Os crawlers podem detectar alteraçõe s nos esquemas da tabela, atualizar as definições da tabela e registrar novas partições conforme novos dados ficam disponíveis. Para obter mais informaçõ es, consulte Propriedades do crawler.	10 de maio de 2019

Compatibilidade com opções adicionais de configuração para trabalhos com uso intenso de memória	Adicionadas informaçõ es sobre as opções de configuração para tarefas do Apache Spark com cargas de trabalho com uso intensivo de memória. Você pode escolher entre duas novas configurações que fornecem mais memória por executor. Para obter mais informações, consulte Adição de trabalhos no AWS Glue.	5 de abril de 2019
Compatibilidade com classific adores personalizados de CSV	Adicionadas informações sobre como usar um classific ador CSV personalizado para inferir o esquema de vários tipos de dados CSV. Para obter mais informaçõ es, consulte Escrever classific adores personalizados.	26 de março de 2019
Support para tags AWS de recursos	Foram adicionadas informaçõ es sobre o uso de tags de AWS recursos para ajudar você a gerenciar e controlar o acesso aos seus AWS Glue recursos. Você pode atribuir tags AWS de recursos a trabalhos, gatilhos, endpoints e rastreadores no. AWS Glue Para obter mais informaçõ es, consulte Tags da AWS no AWS Glue.	20 de março de 2019

Compatibilidade do Data Catalog com trabalhos do Spark SQL	Foram adicionadas informações sobre como configurar seus AWS Glue trabalhos e endpoints de desenvolvimento para usá-los AWS Glue Data Catalog como um Apache Hive Metastore externo. Isso permite que os trabalhos e os endpoints de desenvolvimento executem consultas do Apache Spark SQL diretamente nas tabelas armazenadas no AWS Glue Data Catalog. Para obter mais informações, consulte Suporte do AWS Glue Data Catalog para trabalhos do Spark SQL.	14 de março de 2019
Compatibilidade com trabalhos de shell do Python	Adição de informações sobre trabalhos de shell do Python e o novo campo Maximum capacity (Capacidade máxima). Para obter mais informações, consulte Adicionar trabalhos de shell do Python no AWS Glue.	18 de janeiro de 2019

Compatibilidade com notificações quando houver alterações em bancos de dados e tabelas	Adição de informações sobre eventos que são gerados para alterações em chamadas de API de partição, tabela e banco de dados. Você pode configurar ações em CloudWatch Eventos para responder a esses eventos. Para obter mais informações, consulte Automatização AWS Glue com CloudWatch eventos.	16 de janeiro de 2019
Compatibilidade com criptografia de senhas de conexão	Adicionadas informações sobre criptografia de senhas usadas em objetos de conexão. Para mais informações, consulte Como criptografar senhas de conexão.	11 de dezembro de 2018
Compatibilidade com permissão por recurso e políticas baseadas em recursos	Adição de informações sobre como usar políticas de permissões em nível de recurso e baseadas em recursos com o AWS Glue. Para obter mais informações, consulte os tópicos em Segurança no AWS Glue.	15 de outubro de 2018
Support para SageMaker notebooks	Foram adicionadas informações sobre o uso de SageMaker notebooks com endpoints AWS Glue de desenvolvimento. Para obter mais informações, consulte Gerenciamento de cadernos.	5 de outubro de 2018

Compatibilidade com criptografia	Adicionadas informações sobre como usar criptografia com o AWS Glue. Para obter mais informações, consulte Criptografia em repouso, Criptografia em trânsito e Configurar a criptografia no AWS Glue.	24 de agosto de 2018
Compatibilidade com métricas de trabalho do Apache Spark	Adição de informações sobre o uso de métricas do Apache Spark para melhorar a depuração e a criação de perfis de trabalhos de ETL. Você pode facilment e acompanhar métricas do runtime como bytes lidos e gravados, uso de memória e carga de CPU do driver e executores, e ordem aleatória de dados entre os executore s no console do AWS Glue. Para obter mais informaçõ es, consulte Monitoramento AWS Glue usando CloudWatc h métricas, Monitoramento e depuração de tarefas e Trabalho com tarefas no console. AWS Glue	13 de julho de 2018

Compatibilidade com o DynamoDB como uma origem dos dados	Informações adicionadas sobre o crawling do DynamoDB e como usá-lo como uma origem dos dados de trabalhos de ETL. Para obter mais informações, consulte Catalogamento de tabelas com um crawler e Parâmetros de conexão.	10 de julho de 2018
Atualizações do procedimento para criar um servidor de cadernos	Informações atualizadas sobre como criar um servidor de cadernos em uma instância do Amazon EC2 associada a um endpoint de desenvolvimento. Para obter mais informações, consulte Criação de um servidor de caderno associado a um endpoint de desenvolvimento.	9 de julho de 2018
Atualizações agora disponíveis em RSS	Agora, você pode assinar um feed RSS para receber notificações sobre atualizações do Guia do desenvolvedor do AWS Glue .	25 de junho de 2018
Compatibilidade com notificações de atraso para trabalhos	Adicionadas informações sobre como configurar um limite de atraso quando um trabalho for executado. Para obter mais informações, consulte Adição de trabalhos no AWS Glue.	25 de maio de 2018

Configurar um crawler para acrescentar novas colunas	Foram adicionadas informaçõ es sobre a nova opção de configuração para rastreado res,. MergeNewColumns Para obter mais informações, consulte Configuração de um crawler.	7 de maio de 2018
Compatibilidade com tempo limite para trabalhos	Adicionadas informações sobre como definir um tempo limite quando um trabalho é executado. Para obter mais informações, consulte Adição de trabalhos no AWS Glue.	10 de abril de 2018
Compatibilidade com script de ETL Scala e acionamen to de trabalhos com base em estados adicionais de execução	Informações adicionadas sobre o uso do Scala como linguagem de programação de ETL. Além disso, a API do gatilho agora é compatíve l com disparos quando as condições são atendidas (junto com todas as outras condições). Os trabalhos podem ser acionados com base em uma execução de trabalho com falha ou interrompida (além de uma execução de trabalho concluída com sucesso).	12 de janeiro de 2018

Atualizações anteriores

A tabela a seguir descreve as alterações importantes em cada versão do Guia de desenvolvedor do AWS Glue antes de janeiro de 2018.

Alteração	Descrição	Data
Suporte a fontes de dados XML e nova opção de configuração do crawler	Informações adicionadas sobre a classificação de fontes de dados XML e nova opção de crawler para alterações de partição.	16 de novembro de 2017
Novas transformações, suporte a mecanismos de banco de dados adicionais do Amazon RDS e aprimoramentos do endpoint de desenvolvimento	Informações adicionadas sobre as transformações de filtro e mapa, suporte para o Amazon RDS Microsoft SQL Server e Amazon RDS Oracle, além de novos recursos para endpoints de desenvolvimento.	29 de setembro de 2017
Versão inicial do AWS Glue	Este é o lançamento inicial do Guia do desenvolvedor do AWS Glue .	14 de agosto de 2017

AWS Glossário

Para obter a AWS terminologia mais recente, consulte o AWS glossário na Glossário da AWS Referência.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.