Guide de l'utilisateur

Amazon Athena

Amazon Athena: Guide de l'utilisateur

Table of Contents

Qu'est-ce que Amazon Athena ?

Amazon Athena est un service de requêtes interactif qui facilite l'analyse de données directe dans Amazon Simple Storage Service (Amazon S3) via la syntaxe SQL standard. En effectuant quelques actions AWS Management Console, vous pouvez pointer Athena vers vos données stockées dans Amazon S3 et commencer à utiliser le SQL standard pour exécuter des requêtes ad hoc et obtenir des résultats en quelques secondes.

Pour plus d'informations, consultez Mise en route.

Amazon Athena facilite également l'exécution interactive d'analyses de données à l'aide d'Apache Spark sans avoir à planifier, configurer ou gérer les ressources. Lorsque vous exécutez des applications Apache Spark sur Athena, vous soumettez du code Spark pour traitement et recevez directement les résultats. Utilisez l'expérience simplifiée du bloc-notes dans la console Amazon Athena pour développer des applications Apache Spark en utilisant Python ou API de bloc-notes Athena.

Pour plus d'informations, consultez Démarrage avec Apache Spark sur Amazon Athena.

Athena SQL et Apache Spark fonctionnent sans serveur sur Amazon Athena, ce qui signifie qu'il n'y a aucune infrastructure à mettre en place ou à gérer et que vous ne payez que pour les requêtes que vous exécutez. Athena se met automatiquement à l'échelle en exécutant les requêtes en parallèle, ce qui permet d'obtenir des résultats rapides, même avec des jeux de données volumineux et des requêtes complexes.

Rubriques

Quand utiliser Athena ?

Les services de requête tels qu'Amazon Athena, les entrepôts de données comme Amazon Redshift et les infrastructures de traitement de données sophistiquées comme Amazon EMR répondent tous à différents besoins et cas d'utilisation. Les indications suivantes peuvent vous aider à choisir un ou plusieurs services en fonction de vos besoins.

Amazon Athena

Athena vous aide à analyser les données non structurées, semi-structurées et structurées, stockées dans Simple Storage Service (Amazon S3). Par exemple, des formats de données CSV ou JSON, ou des formats en colonnes, tels qu'Apache Parquet et Apache ORC. Vous pouvez utiliser Athena pour exécuter des requêtes ad hoc en utilisant ANSI SQL, sans avoir besoin d'agréger ou de charger les données dans Athena.

Athena s'intègre à Amazon QuickSight pour faciliter la visualisation des données. Vous pouvez utiliser Athena pour générer des rapports ou explorer les données à l'aide d'outils de business intelligence ou de clients SQL, connectés via un pilote JDBC ou ODBC. Pour plus d'informations, consultez What is Amazon QuickSight dans le guide de QuickSight l'utilisateur Amazon etConnexion à Amazon Athena avec les pilotes ODBC et JDBC.

Athena s'intègre au AWS Glue Data Catalog, qui propose un stockage de métadonnées permanent pour vos données dans Amazon S3. Cela vous permet de créer des tables et de demander des données dans Athena sur la base d'un magasin de métadonnées central disponible sur votre compte Amazon Web Services et intégré aux fonctionnalités ETL et de découverte de données de. AWS Glue Pour plus d'informations, consultez les rubriques Intégration dans AWS Glue et Présentation de AWS Glue du Guide du développeur AWS Glue .

Amazon Athena facilite l'exécution de requêtes interactives sur des données dans Simple Storage Service (Amazon S3) sans avoir à formater les données ni à gérer l'infrastructure. Par exemple, Athena est utile si vous souhaitez exécuter une requête rapide sur les journaux Web pour résoudre un problème de performances sur votre site. Avec Athena, vous pouvez commencer rapidement : il vous suffit de définir une table pour vos données et de commencer à interroger avec SQL standard.

Vous devez utiliser Amazon Athena si vous souhaitez exécuter des requêtes SQL ad hoc interactives sur des données sur Simple Storage Service (Amazon S3), sans avoir à gérer d'infrastructure ou de clusters. Amazon Athena offre le moyen le plus simple d'exécuter des requêtes ad hoc pour des données dans Simple Storage Service (Amazon S3) sans avoir besoin de configurer ou de gérer aucun serveur.

Pour une liste des éléments Services AWS qu'Athena utilise ou auxquels elle s'intègre, consultez. the section called “Service AWS intégrations avec Athena”

Amazon EMR

Amazon EMR rend simple et rentable l'exécution de cadres de traitement hautement distribués tels que Hadoop, Spark et Presto par rapport aux déploiements sur site. Amazon EMR est flexible : vous

pouvez exécuter des applications et du code personnalisés, et définir des paramètres de calcul, de mémoire, de stockage et d'application spécifiques pour optimiser vos besoins analytiques.

Outre l'exécution de requêtes SQL, Amazon EMR peut exécuter une grande variété de tâches de traitement de données en mode scale-out (évolutivité horizontale) pour des applications telles que le machine learning, l'analytique de graphes, la transformation de données, le streaming de données et pratiquement tout ce que vous pouvez coder. Vous devriez utiliser Amazon EMR si vous utilisez du code personnalisé pour traiter et analyser des jeux de données extrêmement volumineux avec les derniers cadres de traitement big data tels que Spark, Hadoop, Presto ou Hbase. Amazon EMR vous donne le contrôle total de la configuration de vos clusters et des logiciels installés sur ceux-ci.

Vous pouvez utiliser Amazon Athena pour interroger les données que vous traitez à l'aide d'Amazon EMR. Amazon Athena prend en charge la plupart des mêmes formats de données qu'Amazon EMR. Le catalogue de données d'Athena est compatible avec les metastores Hive. Si vous utilisez EMR et que vous possédez déjà un metastore Hive, vous pouvez exécuter vos instructions DDL sur Amazon Athena et interroger vos données immédiatement sans affecter vos tâches Amazon EMR.

Amazon Redshift

Un entrepôt de données comme Amazon Redshift est votre meilleur choix lorsque vous devez rassembler des données provenant de nombreuses sources différentes (comme les systèmes d'inventaire, les systèmes financiers et les systèmes de vente au détail) dans un format commun et les stocker pendant de longues périodes. Si vous souhaitez créer des rapports métier sophistiqués à partir de données historiques, un entrepôt de données tel qu'Amazon Redshift est le meilleur choix. Le moteur de requête d'Amazon Redshift a été optimisé pour fonctionner particulièrement bien sur l'exécution de requêtes complexes qui rejoignent un grand nombre de tables de bases de données très volumineuses. Lorsque vous devez exécuter des requêtes sur des données hautement structurées avec de nombreuses jointures sur de nombreuses tables très volumineuses, choisissez Amazon Redshift.

Pour plus d'informations sur le moment où il convient d'utiliser Athena, consultez les ressources suivantes :

- Guide de décision pour les services d'analyse sur AWS dans Premiers pas avec le Centre de ressources
- Quand utiliser Athena par rapport à d'autres services de big data dans les FAQ Amazon Athena
- Présentation d'Amazon Athena
- Fonctions Amazon Athena

- FAQ Amazon Athena
- Articles du blog Amazon Athena

Service AWS intégrations avec Athena

Vous pouvez utiliser Athena pour interroger les données Services AWS répertoriées dans cette section. Pour voir les régions prises en charge pour chaque service, consultez Régions et points de terminaison dans le Référence générale d'Amazon Web Services.

Services AWS intégré à Athena

- AWS CloudFormation
- Amazon CloudFront
- AWS CloudTrail
- Amazon DataZone
- Elastic Load Balancing
- Amazon EMR Studio
- AWS Glue Data Catalog
- AWS Identity and Access Management (IAM)
- Amazon QuickSight
- Inventaire Simple Storage Service (Amazon S3)
- AWS Step Functions
- Inventaire AWS Systems Manager
- Amazon Virtual Private Cloud

Pour plus d'informations sur chaque intégration, consultez les sections suivantes.

AWS CloudFormation

Réservation de capacité

Rubrique de référence AWS: ::Athena : : CapacityReservation dans le guide de l'utilisateur AWS CloudFormation

Spécifie une réserve de capacité avec le nom et le nombre fournis d'unités de traitement de données demandées. Pour plus d'informations, consultez Gestion de la capacité de

traitement des requêtes le guide de l'utilisateur Amazon Athena et le manuel de référence CreateCapacityReservationdes API Amazon Athena.

Data catalog (Catalogue de données)

Rubrique de référence AWS: ::Athena : : DataCatalog dans le guide de l'utilisateur AWS CloudFormation

Spécifiez un catalogue de données Athena, y compris un nom, une description, un type, des paramètres et des étiquettes. Pour plus d'informations, consultez Présentation des tables, des bases de données et des catalogues de données le guide de l'utilisateur Amazon Athena et le manuel de référence CreateDataCatalogdes API Amazon Athena.

Requête nommée

Rubrique de référence AWS: ::Athena : : NamedQuery dans le guide de l'utilisateur AWS CloudFormation

Spécifiez des requêtes nommées avec AWS CloudFormation Athena et exécutez-les dans Athena. Les requêtes nommées vous permettent de mapper un nom de requête en requête, puis de l'exécuter en tant que requête enregistrée à partir de la console Athena. Pour plus d'informations, consultez Utilisation de requêtes enregistrées le guide de l'utilisateur Amazon Athena et le manuel de référence CreateNamedQuerydes API Amazon Athena.

Instruction préparée

Rubrique de référence AWS: ::Athena : : PreparedStatement dans le guide de l'utilisateur AWS CloudFormation

Spécifie une instruction préparée à utiliser avec des requêtes SQL dans Athena. Une instruction préparée contient des paramètres dont les valeurs sont fournies au moment de l'exécution. Pour plus d'informations, consultez Utilisation de requêtes paramétrées le guide de l'utilisateur Amazon Athena et le manuel de référence CreatePreparedStatementdes API Amazon Athena.

WorkGroup

Rubrique de référence AWS: ::Athena : : WorkGroup dans le guide de l'utilisateur AWS CloudFormation

Spécifiez les groupes de travail Athena à l'aide de. AWS CloudFormation Utilisez des groupes de travail Athena pour isoler vos requêtes ou celles de votre groupe des autres requêtes dans le même compte. Pour plus d'informations, consultez Utilisation des groupes de travail pour

contrôler l'accès aux requêtes et les coûts le guide de l'utilisateur Amazon Athena et le manuel de référence CreateWorkGroupdes API Amazon Athena.

Amazon CloudFront

Thème de référence : Interrogation des journaux Amazon CloudFront

Utilisez Athena pour interroger les journaux Amazon CloudFront . Pour plus d'informations sur l'utilisation CloudFront, consultez le manuel Amazon CloudFront Developer Guide.

AWS CloudTrail

Thème de référence : Journaux d'interrogation AWS CloudTrail

L'utilisation d'Athena avec CloudTrail les journaux est un moyen efficace d'améliorer votre analyse de l'activité des AWS services. Par exemple, vous pouvez utiliser des requêtes pour identifier des tendances et isoler davantage l'activité par attribut, comme l'adresse IP source ou l'utilisateur. Vous pouvez créer des tables pour interroger les journaux directement depuis la CloudTrail console et utiliser ces tables pour exécuter des requêtes dans Athena. Pour plus d'informations, consultez Utilisation de la CloudTrail console pour créer une table Athena pour les journaux CloudTrail .

Amazon DataZone

Thème de référence : Utilisation d'Amazon DataZone dans Athena

Utilisez Amazon DataZone pour partager, rechercher et découvrir des données à grande échelle au-delà des frontières de l'organisation. DataZonesimplifie votre expérience grâce à des services AWS d'analyse tels qu'Athena AWS Glue, et. AWS Lake Formation Si vous disposez de grandes quantités de données provenant de différentes sources de données, vous pouvez utiliser Amazon DataZone pour créer des groupes de personnes, de données et d'outils basés sur des cas d'utilisation professionnels.

Dans Athena, vous pouvez utiliser l'éditeur de requêtes pour accéder aux environnements et les interroger DataZone. Pour plus d'informations, consultez Utilisation d'Amazon DataZone dans Athena.

Elastic Load Balancing

Thème de référence : Interrogation des journaux de l'Application Load Balancer

L'interrogation des journaux de l'Application Load Balancer vous permet de connaître la source du trafic et la latence, ainsi que les octets transférés vers et depuis les instances Elastic Load

Balancing et les applications backend. Pour plus d'informations, consultez Interrogation des journaux de l'Application Load Balancer.

Thème de référence : Interrogation des journaux de Classic Load Balancer

Interrogez les journaux de Classic Load Balancer pour analyser et comprendre les modèles de trafic en direction et en provenance des instances Elastic Load Balancing et des applications backend. Vous pouvez voir la source du trafic, la latence et les octets transférés. Pour plus d'informations, consultez la rubrique Création de la table pour les journaux ELB.

Amazon EMR Studio

Rubrique de référence : Use the Amazon Athena SQL editor in EMR Studio

Vous pouvez utiliser Athena dans un EMR Studio pour développer et exécuter des requêtes interactives. Cela vous permet d'utiliser EMR Studio pour les analytiques SQL sur Athena à partir de la même interface Amazon EMR que celle que vous utilisez pour vos charges de travail Spark, Scala et autres. Avec l'intégration Athena dans EMR Studio, vous pouvez effectuer les tâches suivantes :

- Exécuter des requêtes Athena SQL
- Afficher les résultats des requêtes
- Afficher l'historique des requêtes
- Afficher les requêtes enregistrées
- Exécuter des requêtes paramétrées
- Afficher les bases de données, les tables et les vues d'un catalogue de données

Les fonctionnalités Athena suivantes ne sont pas disponibles dans Amazon EMR Studio :

- Fonctionnalités administrateur telles que la création ou la mise à jour de groupes de travail, de sources de données ou de réserves de capacité Athena
- Athena pour Spark ou blocs-notes Spark
- DataZone intégration
- Step Functions

L'intégration d'EMR Studio à Athena est disponible partout où Régions AWS EMR Studio et Athena sont disponibles. Pour plus d'informations sur l'utilisation d'Athena dans EMR Studio, consultez la rubrique Use the Amazon Athena SQL editor in EMR Studio dans le Guide de gestion Amazon EMR.

AWS Glue Data Catalog

Thème de référence : Intégration avec AWS Glue

Athena s'intègre au AWS Glue Data Catalog, qui propose un stockage de métadonnées permanent pour vos données dans Amazon S3. Cela vous permet de créer des tables et de demander des données dans Athena sur la base d'un magasin de métadonnées central disponible sur votre compte Amazon Web Services et intégré aux fonctionnalités ETL et de découverte de données de. AWS Glue Pour plus d'informations, consultez Intégration avec AWS Glue et Présentation de AWS Glue dans le Guide du développeur AWS Glue .

AWS Identity and Access Management (JE SUIS)

Rubrique de référence : Actions pour Amazon Athena

Vous pouvez utiliser des actions d'API Athena dans les politiques d'autorisation IAM. Pour plus d'informations, consultez les rubriques Actions pour Amazon Athena et Gestion des identités et des accès dans Athena.

Amazon QuickSight

Thème de référence : Connexion à Amazon Athena avec les pilotes ODBC et JDBC

Athena s'intègre à Amazon QuickSight pour faciliter la visualisation des données. Vous pouvez utiliser Athena pour générer des rapports ou explorer les données à l'aide d'outils de business intelligence ou de clients SQL, connectés via un pilote JDBC ou ODBC. Pour plus d'informations sur Amazon QuickSight, consultez la section Qu'est-ce qu'Amazon QuickSight dans le guide de QuickSight l'utilisateur Amazon. Pour plus d'informations sur l'utilisation des pilotes JDBC et ODBC avec Athena, consultez la rubrique Connexion à Amazon Athena avec les pilotes ODBC et JDBC.

Inventaire Simple Storage Service (Amazon S3)

Rubrique de référence : Interrogation d'un inventaire avec Athena dans le Guide de l'utilisateur Amazon Simple Storage Service

Vous pouvez utiliser Amazon Athena pour interroger l'inventaire Simple Storage Service (Amazon S3) à l'aide du langage SQL standard. Vous pouvez utiliser l'inventaire Simple Storage Service (Amazon S3) pour contrôler et signaler le statut de réplication et de chiffrement de vos objets à des fins professionnelles, de conformité et d'obligations réglementaires. Pour plus d'informations, veuillez consulter la rubrique Inventaire Simple Storage Service (Amazon S3) du Guide de l'utilisateur Amazon Simple Storage Service.

AWS Step Functions

Rubrique de référence : Appel d'Athena avec Step Functionsdans le Guide du développeur AWS Step Functions

Appelle Athéna avec. AWS Step Functions AWS Step Functions peut contrôler la sélection Services AWS directement à l'aide de l'Amazon States Language. Vous pouvez utiliser Step Functions avec Athena pour lancer et arrêter l'exécution de requêtes, obtenir des résultats de requêtes, exécuter des requêtes de données ad hoc ou planifiées et récupérer les résultats des lacs de données dans Simple Storage Service (Amazon S3). Le rôle Step Functions doit être autorisé à utiliser Athena. Pour plus d'informations, consultez le Guide du développeur AWS Step Functions.

Vidéo : orchestrez les requêtes Amazon Athena à l'aide AWS Step Functions

La vidéo suivante montre comment utiliser Amazon Athena, AWS Step Functions exécuter une requête Athena régulièrement planifiée et générer un rapport correspondant.

Orchestrez les requêtes Amazon Athena à l'aide de AWS Step Functions

Pour un exemple qui utilise Step Functions et Amazon EventBridge pour orchestrer AWS Glue DataBrew, Athena et QuickSight Amazon, consultez la section Orchestrating AWS Glue DataBrew an job and Amazon Athena query AWS Step Functions with dans le blog Big Data. AWS

AWS Systems Manager Inventory

Rubrique de référence : Interrogation des données d'inventaire à partir de plusieurs régions et comptes dans le Guide de l'utilisateur AWS Systems Manager

AWS Systems Manager Inventory s'intègre à Amazon Athena pour vous aider à interroger les données d'inventaire provenant de plusieurs comptes Régions AWS . Pour plus d'informations, consultez le Guide de l'utilisateur AWS Systems Manager.

Amazon Virtual Private Cloud

Thème de référence : Interrogation des journaux de flux Amazon VPC

Les journaux de flux de cloud privé virtuel Amazon Virtual Private Cloud capturent des informations sur le trafic IP circulant vers et depuis les interfaces réseau d'un VPC. Interrogez les journaux dans Athena pour examiner les modèles de trafic réseau, et identifier les menaces et les risques au sein de votre réseau Amazon VPC. Pour plus d'informations sur Amazon VPC, consultez le Guide de l'utilisateur Amazon VPC.

Configuration

Si vous êtes déjà inscrit à Amazon Web Services, vous pouvez commencer à utiliser Amazon Athena dès maintenant. Si vous ne vous êtes pas inscrit AWS ou si vous avez besoin d'aide pour démarrer, assurez-vous d'effectuer les tâches suivantes.

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. Pour des raisons de sécurité, attribuez un accès administratif à un utilisateur et utilisez uniquement l'utilisateur root pour effectuer les tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. Vous pouvez afficher l'activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en choisissant Mon compte.

Création d'un utilisateur doté d'un accès administratif

Après vous être inscrit à un Compte AWS, sécurisez Utilisateur racine d'un compte AWS AWS IAM Identity Center, activez et créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

Pour obtenir de l‘aide pour vous connecter en utilisant l‘utilisateur racine, consultez Connexion en tant qu‘utilisateur racine dans le Guide de l‘utilisateur Connexion à AWS .

2. Activez l‘authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, voir Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d’ AWS IAM Identity Center dans le Guide de l’utilisateur AWS IAM Identity Center .

2. Dans IAM Identity Center, accordez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connectez-vous en tant qu'utilisateur disposant d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l‘URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l‘utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribuer l'accès à des utilisateurs supplémentaires

1. Dans IAM Identity Center, créez un ensemble d'autorisations conforme aux meilleures pratiques en matière d'application des autorisations du moindre privilège.

   Pour obtenir des instructions, voir Création d'un ensemble d'autorisations dans le guide de AWS IAM Identity Center l'utilisateur.

2. Affectez des utilisateurs à un groupe, puis attribuez un accès d'authentification unique au groupe.

Pour obtenir des instructions, voir Ajouter des groupes dans le guide de AWS IAM Identity Center l'utilisateur.

Octroi d’un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l’une des options suivantes.

Quel utilisateur a besoin d’un accès programmatique ?	Pour	Par
Identité de la main-d’œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de l’interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour les AWS SDK, les outils et les AWS API, consultez la section Authentification IAM Identity Center dans le Guide de référence AWS des SDK et des outils.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
	aux AWS CLI AWS SDK ou AWS aux API.	
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer les AWS CLI demandes programmatiques adressées aux AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les AWS SDK et les outils, voir Authentifier à l'aide d'informations d'identification à long terme dans le Guide de AWS référence des SDK et des outils. • Pour les AWS API, consultez la section Gestion des clés d'accès pour les utilisateurs IAM dans le guide de l'utilisateur IAM.

Jonction de politiques gérées pour Athena

Les politiques gérées par Athena accordent des autorisations d'utilisation des fonctionnalités d'Athena. Vous pouvez associer ces politiques gérées à un ou plusieurs rôles IAM que les utilisateurs peuvent assumer pour utiliser Athena.

Un rôle IAM est une identité IAM que vous pouvez créer dans votre compte et qui dispose d'autorisations spécifiques. Un rôle IAM est similaire à un utilisateur IAM dans la mesure où il s'agit d'une AWS identité dotée de politiques d'autorisation qui déterminent ce que l'identité peut et ne

peut pas faire. AWS En revanche, au lieu d'être associé de manière unique à une personne, un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. En outre, un rôle ne dispose pas d'informations d'identification standard à long terme comme un mot de passe ou des clés d'accès associées. Au lieu de cela, lorsque vous adoptez un rôle, il vous fournit des informations d'identification de sécurité temporaires pour votre session de rôle.

Pour plus d'informations sur les rôles, voir Rôles IAM et Création de rôles IAM dans le Guide de l'utilisateur IAM.

Pour créer un rôle qui donne accès à Athena, vous devez associer des politiques gérées par Athena au rôle. Il existe deux politiques gérées pour Athena : `AmazonAthenaFullAccess` et `AWSQuicksightAthenaAccess`. Ces politiques accordent des autorisations à Athena pour interroger Simple Storage Service (Amazon S3) et pour écrire les résultats de vos requêtes dans un compartiment séparé en votre nom. Pour afficher le contenu de ces politiques pour Athena, voir AWS politiques gérées pour Amazon Athena.

Pour connaître les étapes permettant d'associer les politiques gérées d'Athena à un rôle, suivez la rubrique Ajout d'autorisations d'identité IAM (console) du Guide de l'utilisateur IAM et ajoutez les politiques gérées `AmazonAthenaFullAccess` et `AWSQuicksightAthenaAccess` au rôle que vous avez créé.

> **Note**
>
> Vous pouvez avoir besoin d'autorisations supplémentaires pour accéder au jeu de données sous-jacent dans Simple Storage Service (Amazon S3). Si vous n'êtes pas le propriétaire du compte ou si vous disposez d'un accès restreint à un compartiment, contactez le propriétaire du compartiment pour qu'il vous accorde l'accès à l'aide d'une politique de compartiment basée sur les ressources, ou contactez votre administrateur de compte pour qu'il vous accorde l'accès à l'aide d'une politique basée sur les rôles. Pour plus d'informations, consultez Accès à Amazon S3 depuis Athena. Si le jeu de données ou les résultats de la requête Athena sont chiffrés, il est possible que vous ayez besoin d'autorisations supplémentaires. Pour plus d'informations, voir Chiffrement au repos.

Accès à Athena

Vous pouvez accéder à Athena à l'aide d'une connexion JDBC ou ODBC AWS Management Console, de l'API Athena, de la CLI Athena, du SDK ou. AWS AWS Tools for Windows PowerShell

- Pour commencer à utiliser Athena SQL avec la console, voir Mise en route.
- Pour commencer à créer des blocs-notes compatibles Jupyter et des applications Apache Spark utilisant Python, consultez. Utilisation d'Apache Spark dans Amazon Athena
- Pour savoir comment utiliser les pilotes JDBC ou ODBC, consultezConnexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.
- Pour utiliser l'API Athena, consultez la Référence d'API Amazon Athena.
- Pour utiliser l'interface de ligne de commande (CLI), installez AWS CLI, puis tapez `aws athena help` sur la ligne de commande pour afficher les commandes disponibles. Pour plus d'informations sur les commandes disponibles, consultez la Référence de la ligne de commande Amazon Athena.
- Pour l'utiliser AWS SDK for Java 2.x, consultez la section Athena de la référence d'AWS SDK for Java 2.x API, les exemples d'Athena Java V2 sur GitHub .com et le manuel du développeur.AWS SDK for Java 2.x
- Pour utiliser le AWS SDK for .NET, consultez l'espace de `Amazon.Athena` noms dans la référence de l'AWS SDK for .NET API, les exemples .NET Athena GitHub sur .com et le manuel AWS SDK for .NET du développeur.
- Pour l'utiliser AWS Tools for Windows PowerShell, consultez la AWS Tools for PowerShell référence de l'applet de commande Amazon Athena, la page du AWS Tools for PowerShellportail et le guide de l'utilisateur.AWS Tools for Windows PowerShell
- Pour de plus amples informations sur les points de terminaison de service Athena auquel vous pouvez vous connecter par programmation, consultez les points de terminaison et quotas Amazon Athena dans le Référence générale d'Amazon Web Services.

Utilisation d'Athena SQL

Vous pouvez utiliser Athena SQL pour interroger vos données existantes dans Amazon S3 à l'aide du AWS Glue Data Catalog, d'un métastore Hive externe ou de requêtes fédérées à l'aide de divers connecteurs prédéfinis vers d'autres sources de données.

Vous pouvez également :

- Vous connecter à des outils de business intelligence et à d'autres applications à l'aide des pilotes JDBC et ODBC d'Athena.
- Interroger les journaux de service AWS.
- Interroger les tables Apache Iceberg, y compris les requêtes de voyage dans le temps et les jeux de données Apache Hudi.
- Interroger les données géospatiales.
- Interrogez à l'aide de l'inférence d'apprentissage automatique d'Amazon SageMaker.
- Interroger en utilisant vos propres fonctions définies par l'utilisateur.
- Accélérer le traitement des requêtes portant sur des tables hautement partitionnées et automatiser la gestion des partitions grâce à la projection des partitions.

Rubriques

- Référence SQL pour Athena
- Résolution des problèmes dans Athena
- Exemples de code

Présentation des tables, des bases de données et des catalogues de données

Dans Athena, les catalogues, les bases de données et les tables sont des conteneurs pour les définitions de métadonnées qui définissent un schéma pour les données sources sous-jacentes.

Athena utilise les termes suivants pour désigner les hiérarchies d'objets de données :

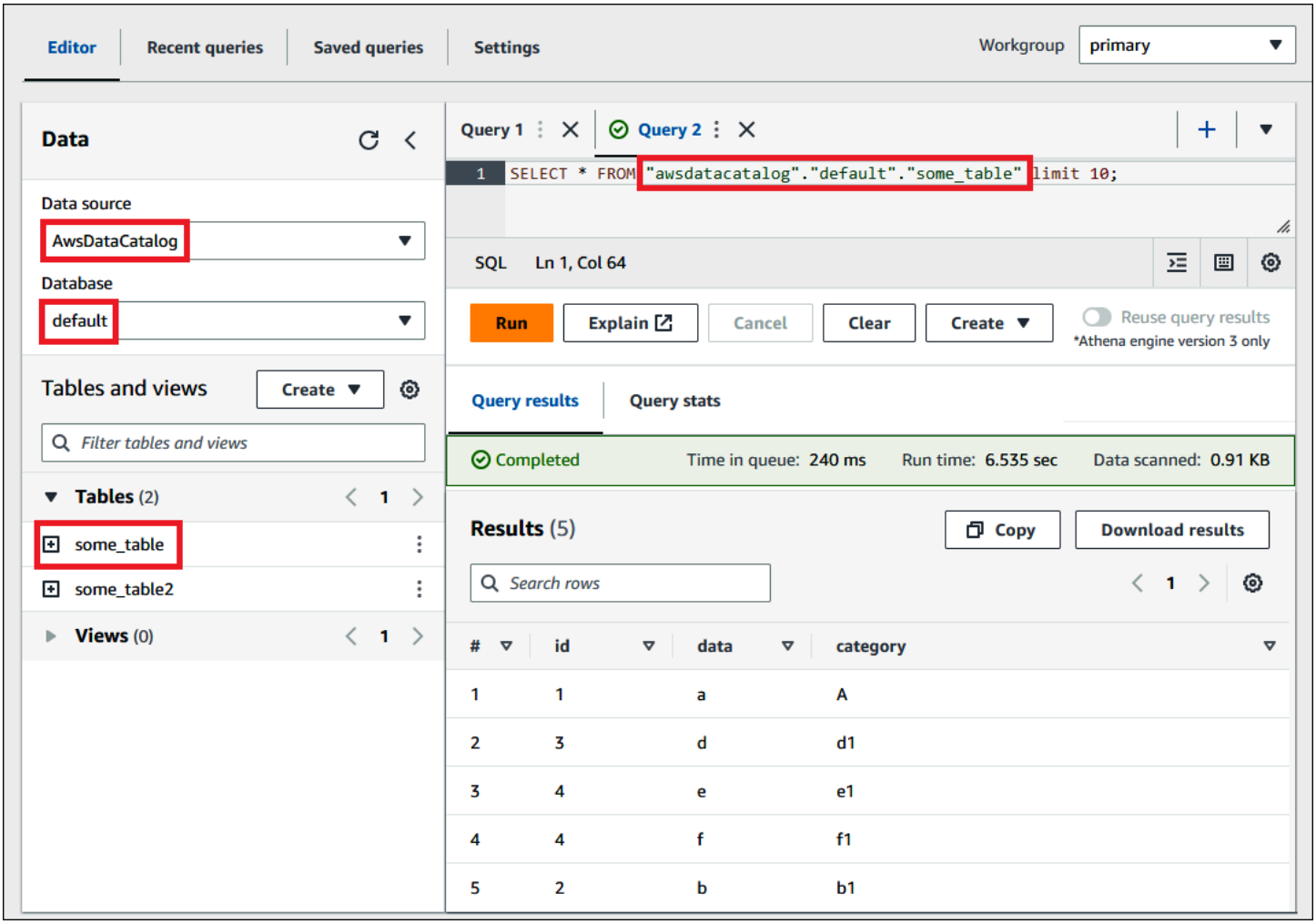
- Source de données : un groupe de bases de données
- Base de données : un groupe de tables
- Table : des données organisées sous la forme d'un groupe de lignes ou de colonnes

Parfois, ces objets sont également désignés par des noms alternatifs mais équivalents, tels que les suivants :

- Une source de données est parfois appelée catalogue.
- Une base de données est parfois appelée schéma.

> **Note**
>
> Cette terminologie peut varier selon les sources de données fédérées que vous utilisez avec Athena. Pour plus d'informations, consultez Athena et qualificateurs de noms de tables fédérées.

L'exemple de requête suivant dans la console Athena utilise la source de données `awsdatacatalog`, la base de données `default` et la table `some_table`.



Il doit y avoir une table dans Athena pour chaque jeu de données. Les métadonnées figurant dans cette table indiquent à Athena où les données sont situées dans Simple Storage Service (Amazon S3) et spécifient la structure des données, telle que les noms de colonne, les types de données et le nom de la table. Les bases de données constituent un regroupement logique de tables et stockent uniquement les métadonnées et les informations de schéma pour un ensemble de données.

Pour chaque jeu de données que vous souhaitez interroger, Athena doit avoir une table sous-jacente qu'il utilisera pour obtenir et renvoyer les résultats de requête. Par conséquent, avant d'exécuter des requêtes sur les données, une table doit être enregistrée dans Athena. L'enregistrement se produit lorsque vous créez des tables automatiquement ou manuellement.

Vous pouvez créer une table automatiquement à l'aide d'un AWS Glue robot d'exploration. Pour plus d'informations sur AWS Glue les robots d'exploration, consultez la section Intégration avec AWS Glue. Lorsqu'il AWS Glue crée une table, elle l'enregistre dans son propre catalogue de AWS Glue données. Athena utilise le catalogue de données AWS Glue pour stocker et récupérer ces

métadonnées, et les utiliser lorsque vous exécutez des requêtes pour analyser le jeu de données sous-jacent.

Quelle que soit la façon dont les tables sont créées, le processus de création des tables enregistre le jeu de données dans Athena. Cet enregistrement a lieu dans le AWS Glue Data Catalog et permet à Athena d'exécuter des requêtes sur les données. Dans l'éditeur de requêtes Athena, ce catalogue (ou source de données) est désigné par l'étiquette `AwsDataCatalog`.

Après avoir créé une table, vous pouvez utiliser les instructions SQL SELECT pour l'interroger, notamment pour obtenir des emplacements de fichiers spécifiques pour vos données sources. Les résultats de votre requête sont stockés dans Simple Storage Service (Amazon S3) dans l'emplacement de résultats de la requête que vous avez spécifié.

Le catalogue de AWS Glue données est accessible via votre compte Amazon Web Services. D'autres Services AWS peuvent partager le catalogue de AWS Glue données, afin que vous puissiez voir les bases de données et les tables créées au sein de votre organisation à l'aide d'Athena et vice versa.

- Pour créer manuellement une table :
  - Utilisez la console Athena pour exécuter l'Assistant de création de table.
  - Utilisez la console Athena pour écrire des instructions DDL Hive dans l'éditeur de requête.
  - Utilisez l'interface de ligne de commande (CLI) ou l'API Athena pour exécuter une chaîne de requête SQL avec des instructions DDL.
  - Utilisez le pilote JDBC ou ODBC Athena.

Lorsque vous créez manuellement des tables et des bases de données, Athena utilise des instructions DDL (Data Definition Language) HiveQL telles que `CREATE TABLE`, `CREATE DATABASE` et `DROP TABLE` en arrière-plan pour créer des tables et des bases de données dans le catalogue de données AWS Glue Data Catalog.

Pour commencer, vous pouvez utiliser un didacticiel dans la console Athena ou consulter un step-by-step guide de la documentation d'Athena.

- Pour utiliser le didacticiel dans la console Athena, cliquez sur l'icône d'information en haut à droite de la console, puis sur l'onglet Didacticiel.
- Pour un step-by-step didacticiel sur la création d'une table et l'écriture de requêtes dans l'éditeur de requêtes Athena, voir. Mise en route

Mise en route

Ce tutoriel vous guide dans l'utilisation d'Amazon Athena pour l'interrogation de données. Vous allez créer une table basée sur des exemples de données stockées dans Amazon Simple Storage Service, interroger la table et vérifier les résultats de la requête.

Le tutoriel utilise des ressources en temps réel. Les requêtes que vous exécutez vous sont donc facturées. Vous ne payez pas pour l'échantillon de données dans l'emplacement que ce tutoriel utilise, mais si vous téléchargez vos propres fichiers de données sur Simple Storage Service (Amazon S3), des frais s'appliquent.

Prérequis

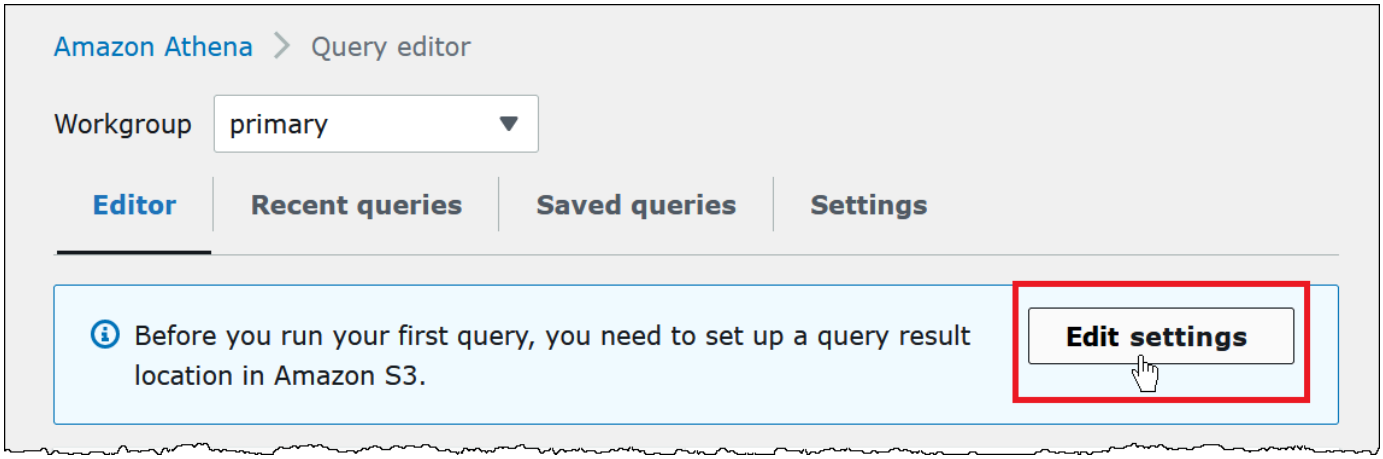
- Si vous ne l'avez pas encore fait, créez un Compte AWS.
- En utilisant le même compte Région AWS (par exemple, US West (Oregon)) et le même compte que celui que vous utilisez pour Athena, suivez les étapes pour créer un compartiment dans Amazon S3 contenant les résultats de vos requêtes Athena. Vous allez configurer ce compartiment pour qu'il soit l'emplacement de sortie de votre requête.

Étape 1 : Créer une base de données

Vous devez d'abord créer une base de données dans Athena.

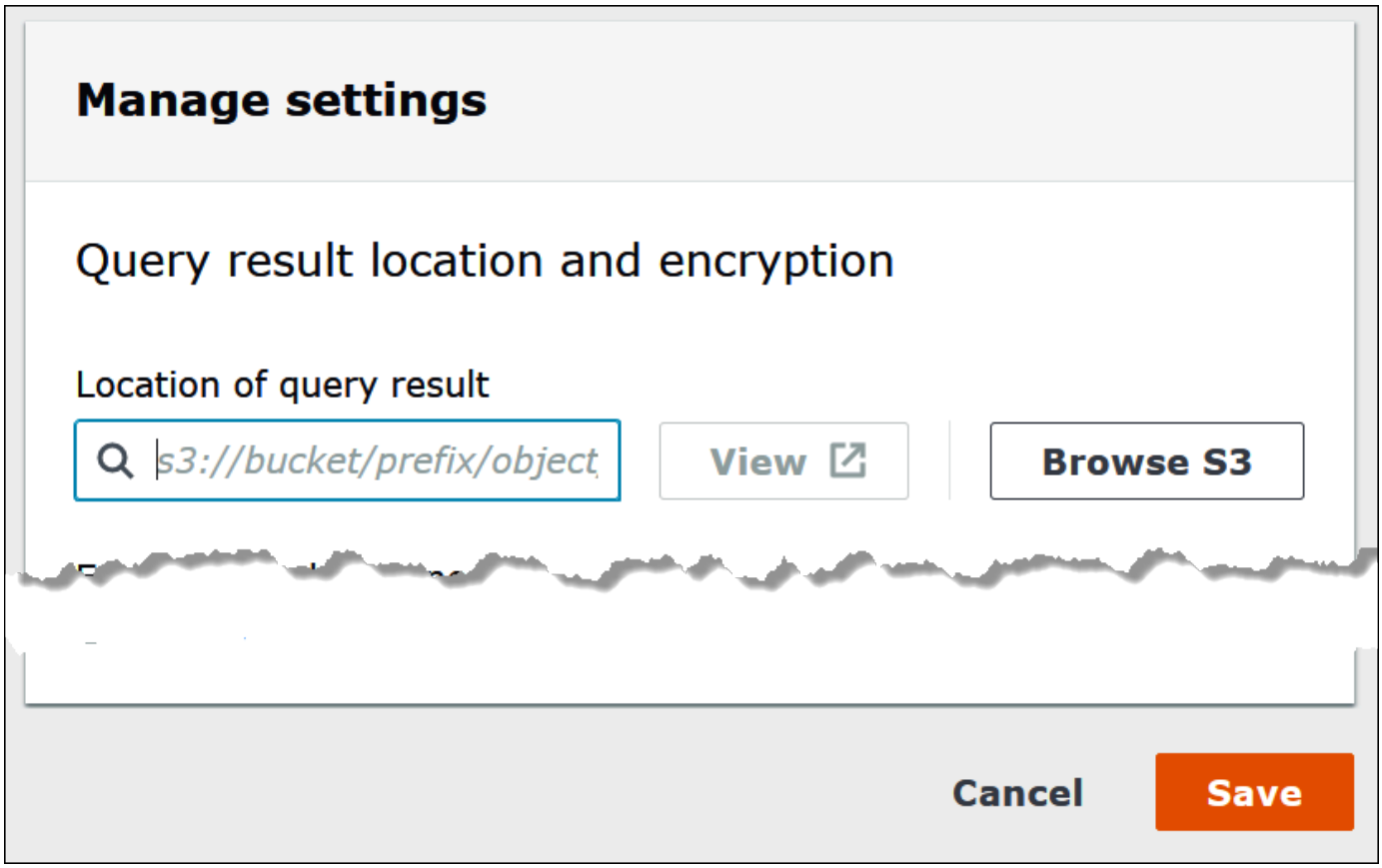
Pour créer une base de données Athena

1. Ouvrez la console Athena à l’adresse https://console.aws.amazon.com/athena/.
2. S'il s'agit de votre première visite sur la console Athena dans votre Région AWS actuelle, choisissez Explore the query editor (Découvrir l'éditeur de requêtes) pour ouvrir l'éditeur de requêtes. Sinon, Athena s'ouvre dans l'éditeur de requêtes.
3. Choisissez Modifier les paramètres pour configurer l'emplacement des résultats de la requête dans Amazon S3.

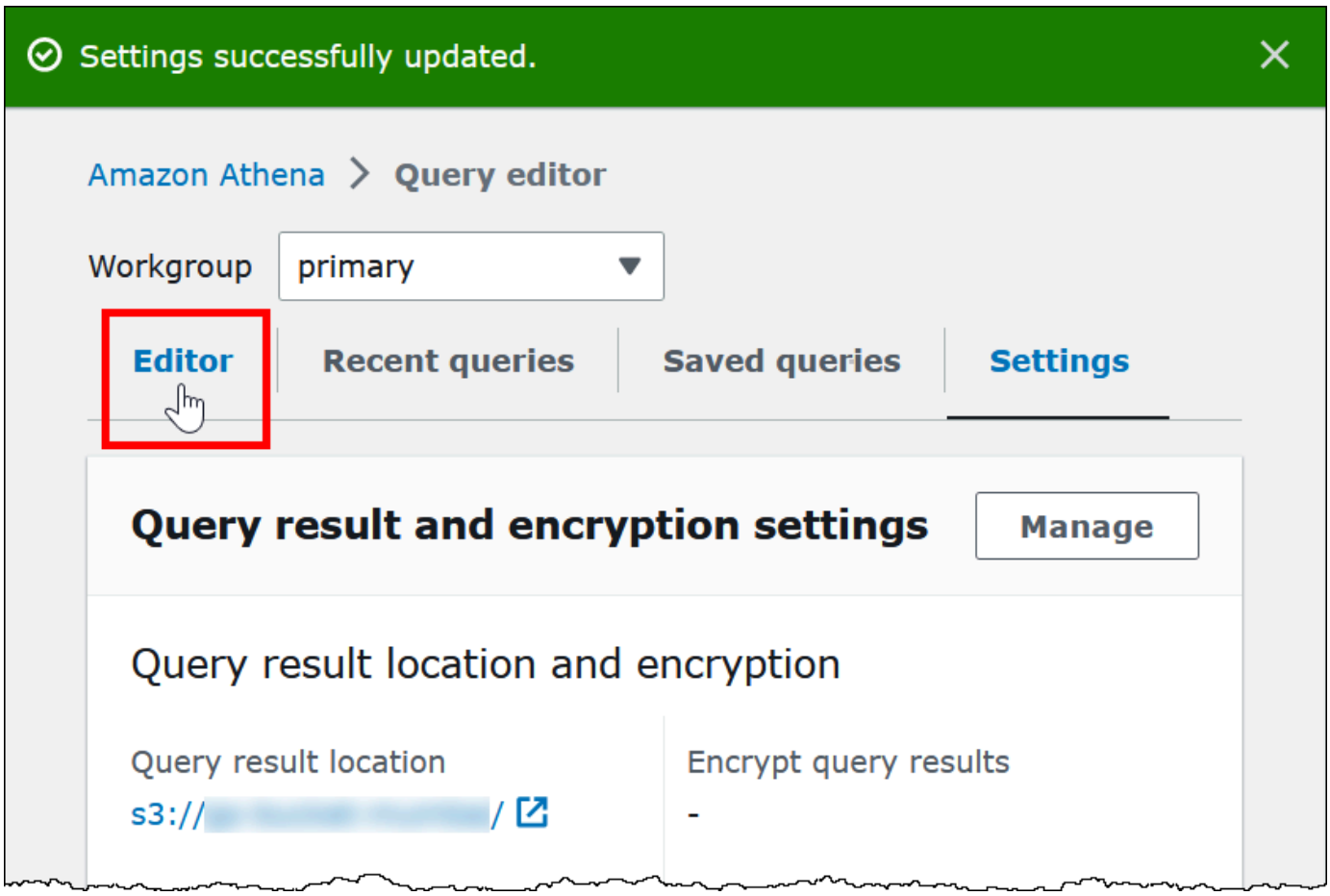


4. Pour Gestion des paramètres, procédez de l'une des manières suivantes :

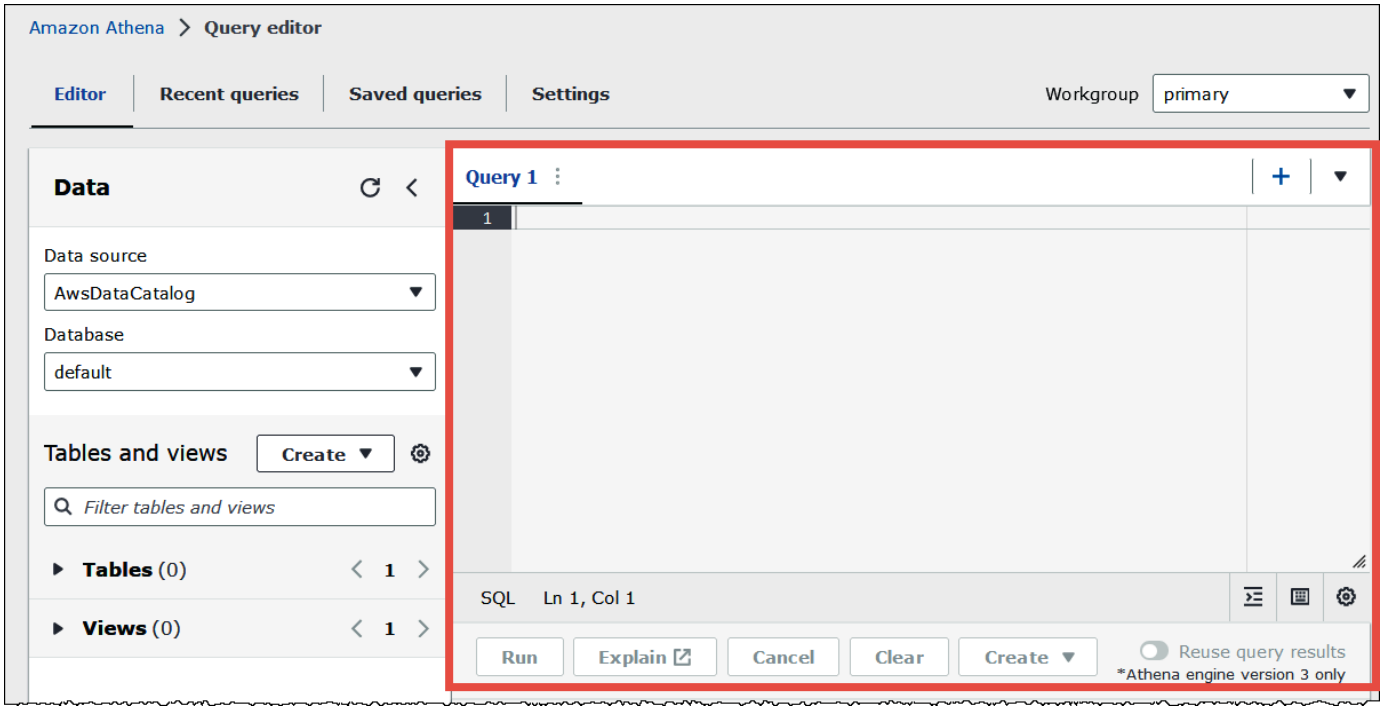
   - Dans la zone de texte Location of query result (Emplacement des résultats de la requête), saisissez le chemin d'accès au compartiment que vous avez créé dans Simple Storage Service (Amazon S3) pour les résultats de votre requête. Préfixez le chemin avec `s3://`.
   - Choisissez Parcourir S3, choisissez le compartiment Simple Storage Service (Amazon S3) que vous avez créé pour votre région actuelle, puis sélectionnez Choisir.



5. Choisissez Enregistrer.
6. Choisissez Editor (Éditeur) pour passer à l'éditeur de requête.



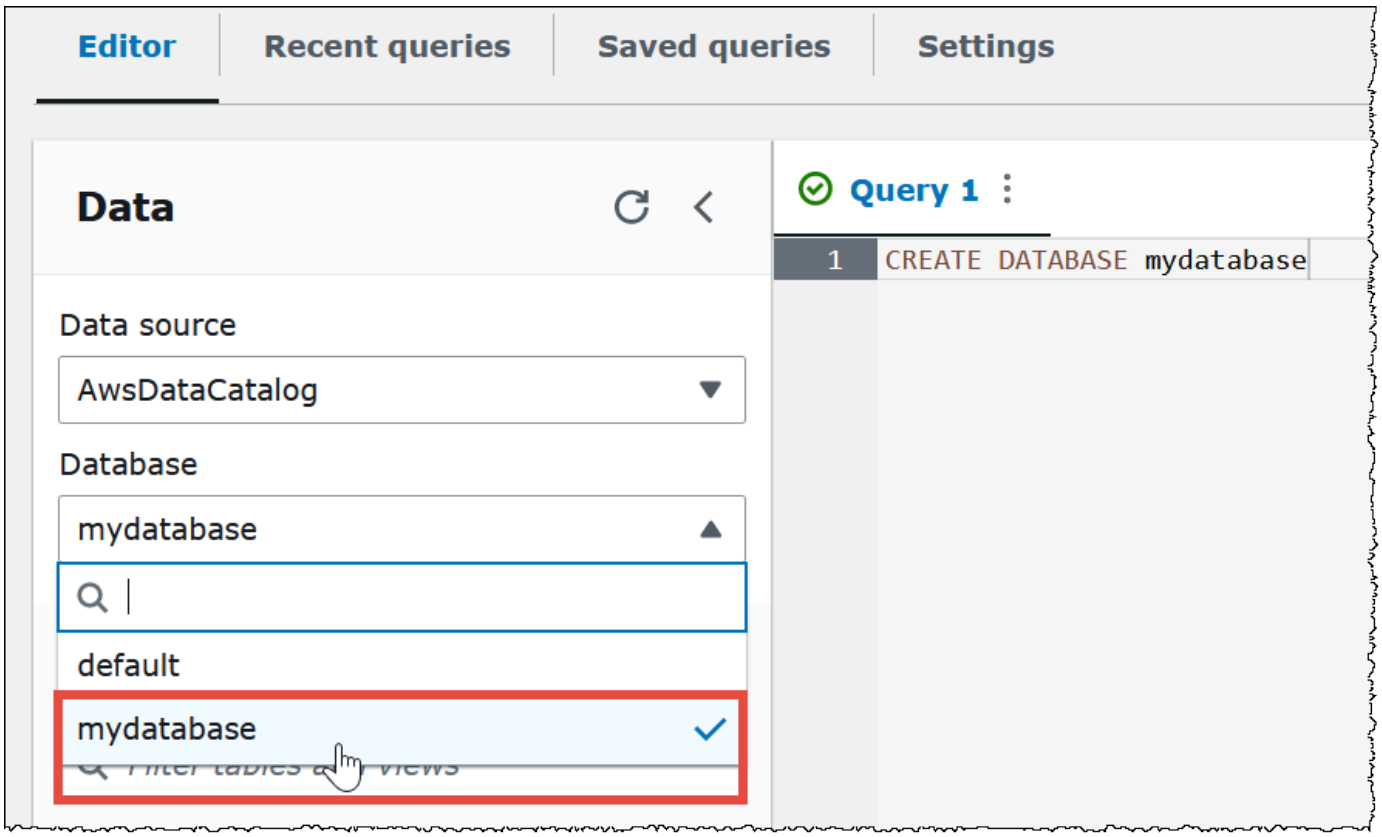
7. À droite du panneau de navigation, vous pouvez utiliser l'éditeur de requête Athena pour saisir et exécuter des requêtes et des instructions.



8. Pour créer une base de données nommée `mydatabase`, saisissez l'instruction CREATE DATABASE suivante.

```
CREATE DATABASE mydatabase
```

9. Choisissez Run (Exécuter) ou appuyez sur **`Ctrl+ENTER`**.
10. Dans la liste des Database (Base de données) à gauche, choisissez `mydatabase` pour en faire votre base de données actuelle.



Étape 2 : créer une table

Maintenant que vous avez une base de données, vous pouvez créer une table Athena pour celle-ci. Le tableau que vous créez sera basé sur des exemples de données de CloudFront journal Amazon dans l'emplacement `s3://athena-examples-`*`myregion`*`/cloudfront/plaintext/` où *`myregion`* est votre emplacement actuel Région AWS.

Les données de l'exemple de journal sont au format TSV (valeurs séparées par des tabulations), ce qui signifie qu'un caractère de tabulation est utilisé comme délimiteur pour séparer les champs. Les données ressemblent à l'exemple suivant. Pour plus de lisibilité, les onglets de l'extrait ont été convertis en espaces et le champ final a été raccourci.

```
2014-07-05 20:00:09 DFW3 4260 10.0.0.15 GET eabcd12345678.cloudfront.net /test-
image-1.jpeg 200 - Mozilla/5.0[...]
2014-07-05 20:00:09 DFW3 4252 10.0.0.15 GET eabcd12345678.cloudfront.net /test-
image-2.jpeg 200 - Mozilla/5.0[...]
2014-07-05 20:00:10 AMS1 4261 10.0.0.15 GET eabcd12345678.cloudfront.net /test-
image-3.jpeg 200 - Mozilla/5.0[...]
```

Pour permettre à Athéna de lire ces données, vous pouvez créer une `CREATE EXTERNAL TABLE` déclaration simple comme celle-ci. L'instruction qui crée la table définit les colonnes qui correspondent aux données, spécifie comment les données sont délimitées et spécifie l'emplacement Simple Storage Service (Amazon S3) qui contient les exemples de données. Notez qu'Athena prévoit d'analyser tous les fichiers d'un dossier, la `LOCATION` clause indique l'emplacement du dossier Amazon S3, et non un fichier spécifique.

N'utilisez pas cet exemple pour l'instant car il comporte une limitation importante qui sera expliquée sous peu.

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_logs (
  `Date` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  ClientInfo STRING
  )
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\t'
  LINES TERMINATED BY '\n'
  LOCATION 's3://athena-examples-my-region/cloudfront/plaintext/';
```

L'exemple crée une table appelée `cloudfront_logs` et spécifie un nom et un type de données pour chaque champ. Ces champs deviennent les colonnes de la table. Comme il `date` s'agit d'un mot réservé, il est masqué par des caractères backtick (`). `ROW FORMAT DELIMITED`signifie qu'Athena utilisera une bibliothèque par défaut appelée LazySimpleSerDepour effectuer le travail réel d'analyse des données. L'exemple spécifie également que les champs sont séparés par des tabulations (`FIELDS TERMINATED BY '\t'`) et que chaque registre dans le fichier se termine par un saut de ligne (`LINES TERMINATED BY '\n`). Enfin, la clause `LOCATION` indique le chemin d'accès dans Simple Storage Service (Amazon S3) où se trouvent les données à lire.

Si vous avez vos propres données séparées par des tabulations ou des virgules, vous pouvez utiliser une `CREATE TABLE` déclaration comme dans l'exemple qui vient d'être présenté, à condition que vos champs ne contiennent pas d'informations imbriquées. Toutefois, si une colonne de `ClientInfo`

ce type contient des informations imbriquées qui utilisent un séparateur différent, une approche différente est requise.

Extraction de données depuis le terrain ClientInfo

En regardant les exemples de données, voici un exemple complet du champ final `ClientInfo` :

```
Mozilla/5.0%20(Android;%20U;%20Windows%20NT%205.1;%20en-US;
%20rv:1.9.0.9)%20Gecko/2009040821%20IE/3.0.9
```

Comme vous pouvez le voir, ce champ est à valeurs multiples. Comme l'exemple d'`CREATE TABLE`instruction qui vient d'être présenté spécifie les onglets comme délimiteurs de champs, il ne peut pas diviser les différents composants du `ClientInfo` champ en colonnes distinctes. Une nouvelle `CREATE TABLE` déclaration est donc requise.

Pour créer des colonnes à partir des valeurs contenues dans le `ClientInfo` champ, vous pouvez utiliser une expression régulière (regex) contenant des groupes d'expressions régulières. Les groupes de regex que vous spécifiez deviennent des colonnes de table distinctes. Pour utiliser une regex dans votre instruction `CREATE TABLE`, utilisez une syntaxe comme la suivante. Cette syntaxe indique à Athena d'utiliser la bibliothèque Régex SerDe et l'expression rationnelle que vous spécifiez.

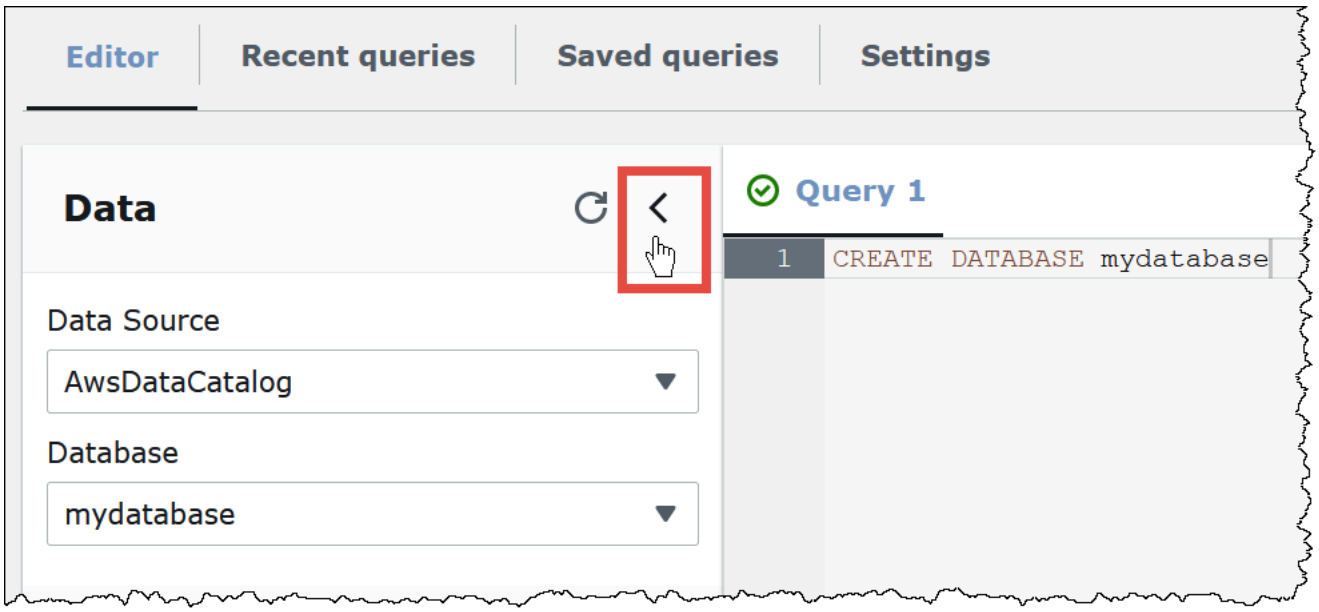
```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
  WITH SERDEPROPERTIES ("input.regex" = "regular_expression")
```

Les expressions rationnelles peuvent être utiles pour créer des tables à partir de données CSV ou TSV complexes, mais elles peuvent être difficiles à écrire et à gérer. Heureusement, il existe d'autres bibliothèques que vous pouvez utiliser pour des formats tels que JSON, Parquet et ORC. Pour plus d'informations, consultez SerDe et formats de données pris en charge.

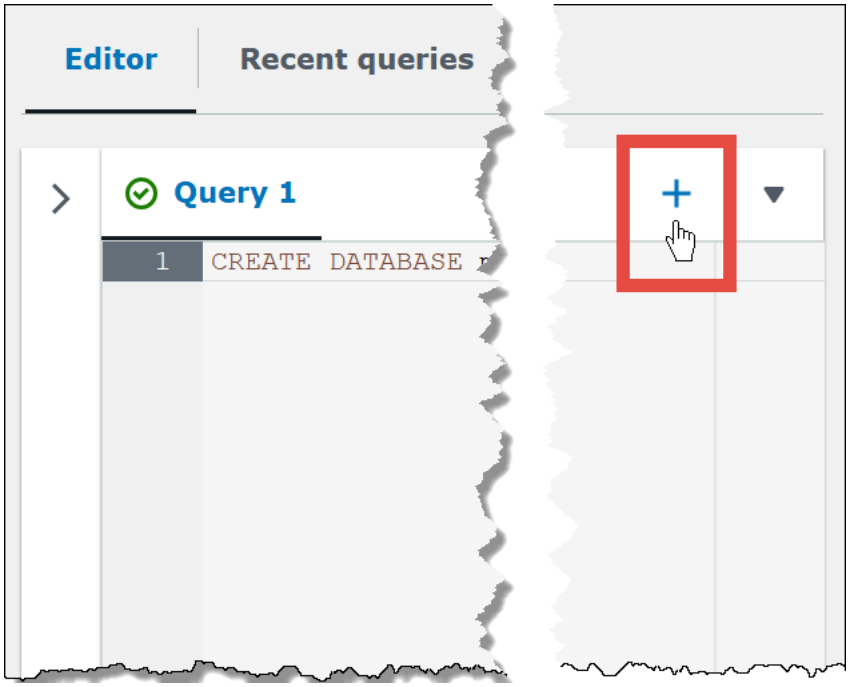
Vous êtes maintenant prêt à créer la table dans l'éditeur de requête Athena. L'instruction `CREATE TABLE` et la regex sont fournies pour vous.

Pour créer une table dans Athena

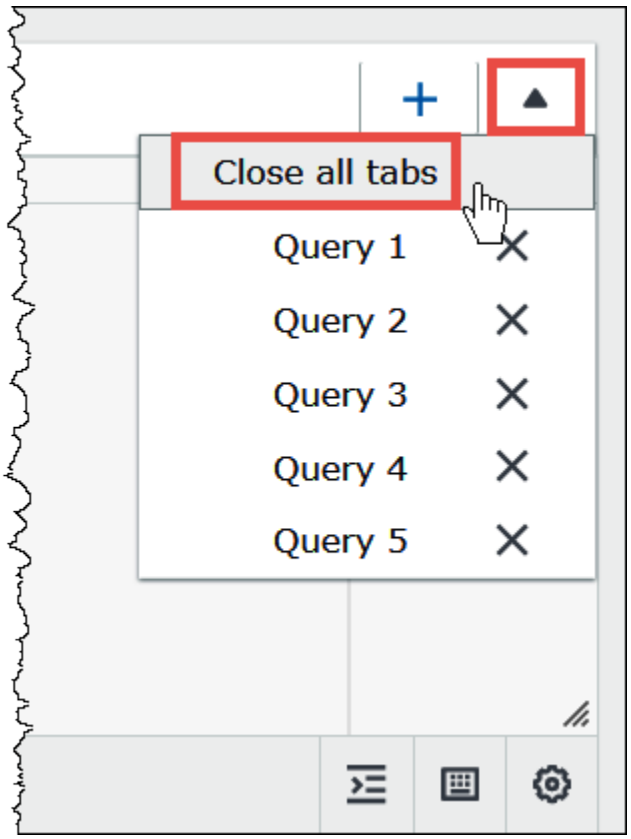
1. Dans le panneau de navigation, pour Database (Base de données), assurez-vous que `mydatabase` est sélectionnée.
2. Pour vous donner plus d'espace dans l'éditeur de requêtes, vous pouvez choisir l'icône de flèche pour réduire le panneau de navigation.



3. Pour créer un onglet pour une nouvelle requête, choisissez le signe plus (+) dans l'éditeur de requête. Vous pouvez ouvrir jusqu'à dix onglets de requête à la fois.



4. Pour fermer un ou plusieurs onglets de requête, choisissez la flèche à côté du signe plus. Pour fermer tous les onglets en même temps, choisissez la flèche, puis choisissez Close all tabs (Fermer tous les onglets).



5. Dans le volet de requête, saisissez l'instruction `CREATE EXTERNAL TABLE` suivante. La regex extrait les informations relatives au système d'exploitation, au navigateur et à la version du navigateur du champ `ClientInfo` des données du journal.

> **Note**
>
> L'expression régulière utilisée dans l'exemple suivant est conçue pour fonctionner avec les données de CloudFront journal d'échantillons accessibles au public sur le site `athena-examples` Amazon S3 et n'est fournie qu'à titre indicatif. Pour d'autres up-to-date expressions régulières qui interrogent à la fois des fichiers CloudFront journaux standard et en temps réel, consultez. Interrogation des journaux Amazon CloudFront

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_logs (
  `Date` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
```

```
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
  BrowserVersion STRING
  )
  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
  WITH SERDEPROPERTIES (
  "input.regex" = "^(?!#)([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s
+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+[^\(]+[\(]([^\;]+).*\%20([^
\/]+)[\/](.*)$"
  ) LOCATION 's3://athena-examples-myregion/cloudfront/plaintext/';
```

6. Dans l'`LOCATION`instruction, remplacez *`myregion`* par celle Région AWS que vous utilisez actuellement (par exemple,`us-west-1`).
7. Cliquez sur Run (Exécuter).

   La table `cloudfront_logs` est créée et apparaît sous la liste Tables pour la base de données `mydatabase`.

Étape 3 : Interrogation des données

Maintenant que la table `cloudfront_logs` a été créée dans Athena à partir des données de Simple Storage Service (Amazon S3), vous pouvez exécuter des requêtes SQL sur la table et en voir les résultats dans Athena. Pour plus d'informations sur l'utilisation de SQL dans Athena, voir Référence SQL pour Athena.

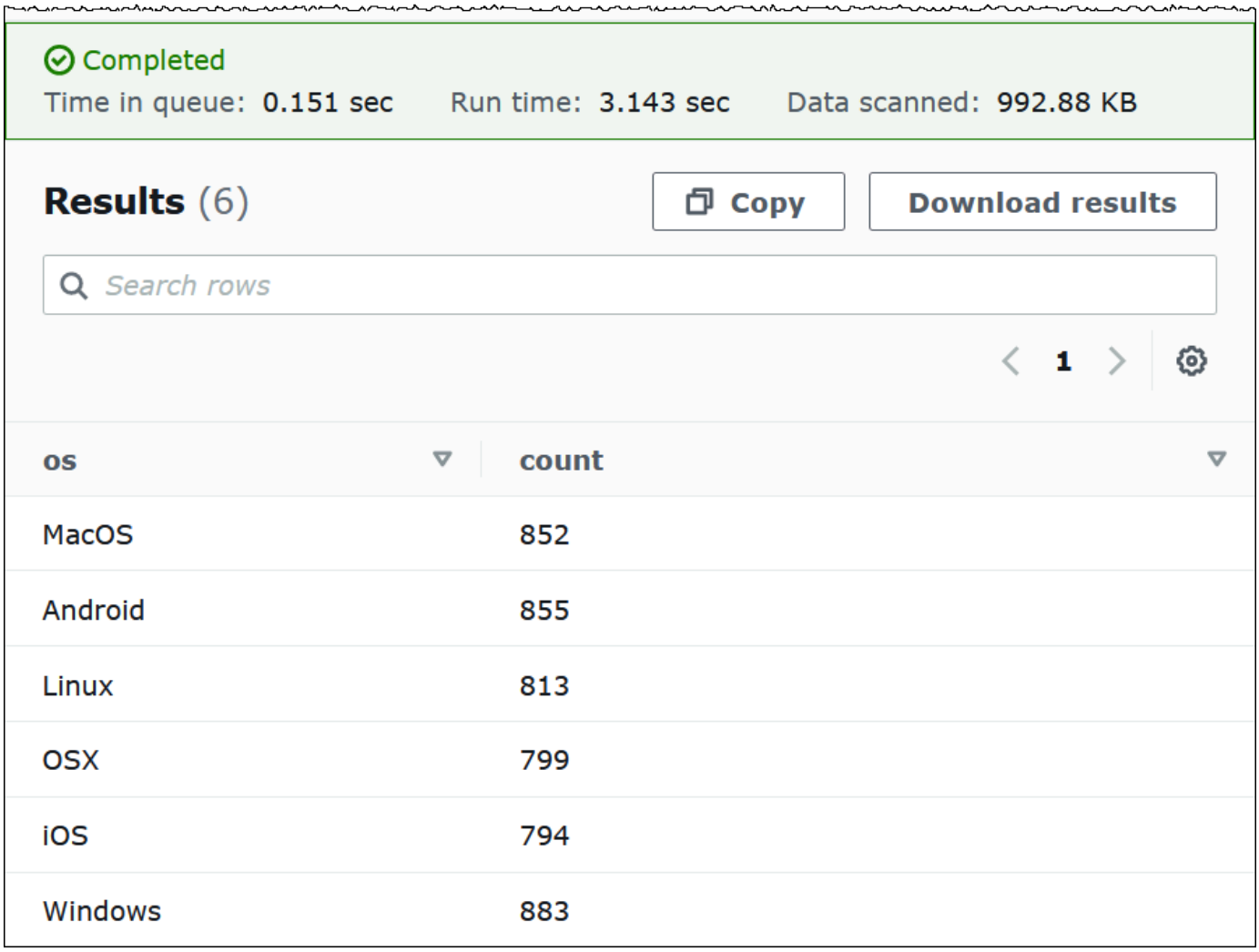
Pour exécuter une requête

1. Choisissez le signe plus (+) pour ouvrir un nouvel onglet de requête et saisissez l'instruction SQL suivante dans le volet de requête.

   ```
   SELECT os, COUNT(*) count
   FROM cloudfront_logs
   WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
   GROUP BY os
   ```

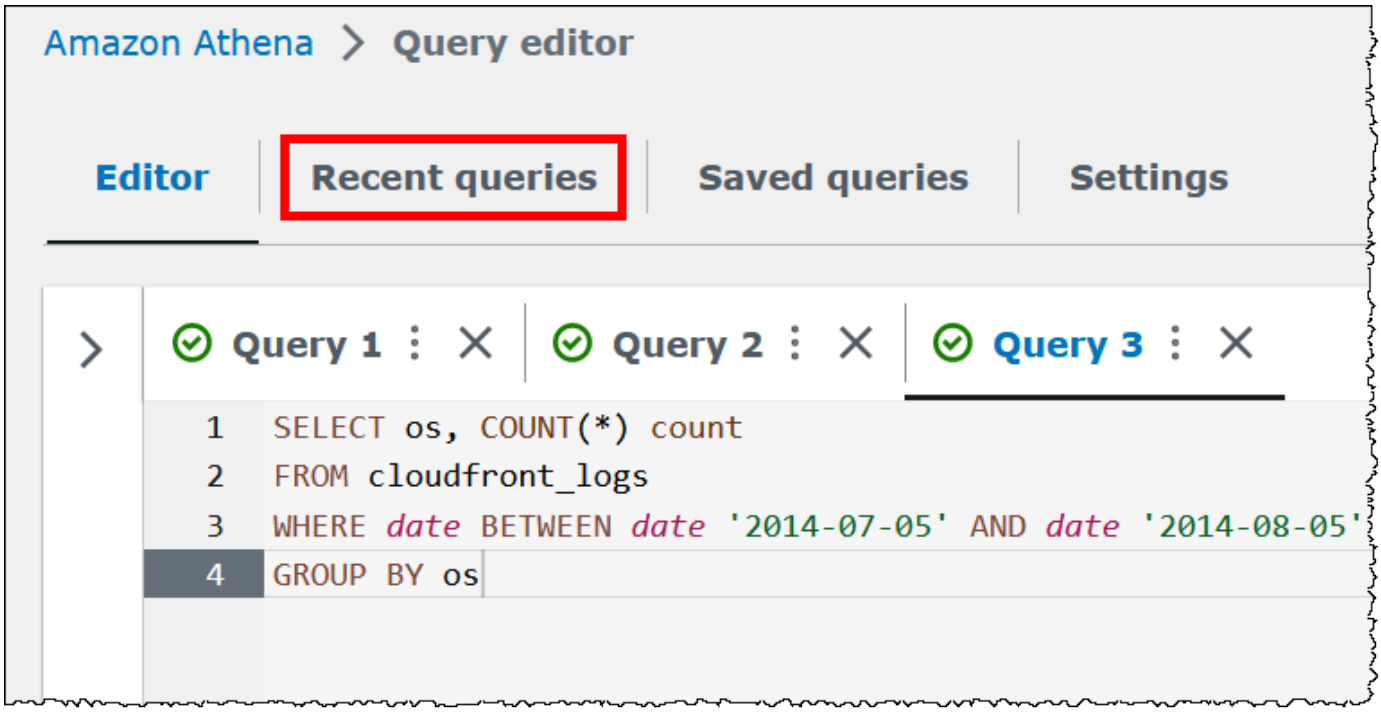
2. Cliquez sur Run (Exécuter).

Les résultats se présentent comme suit :

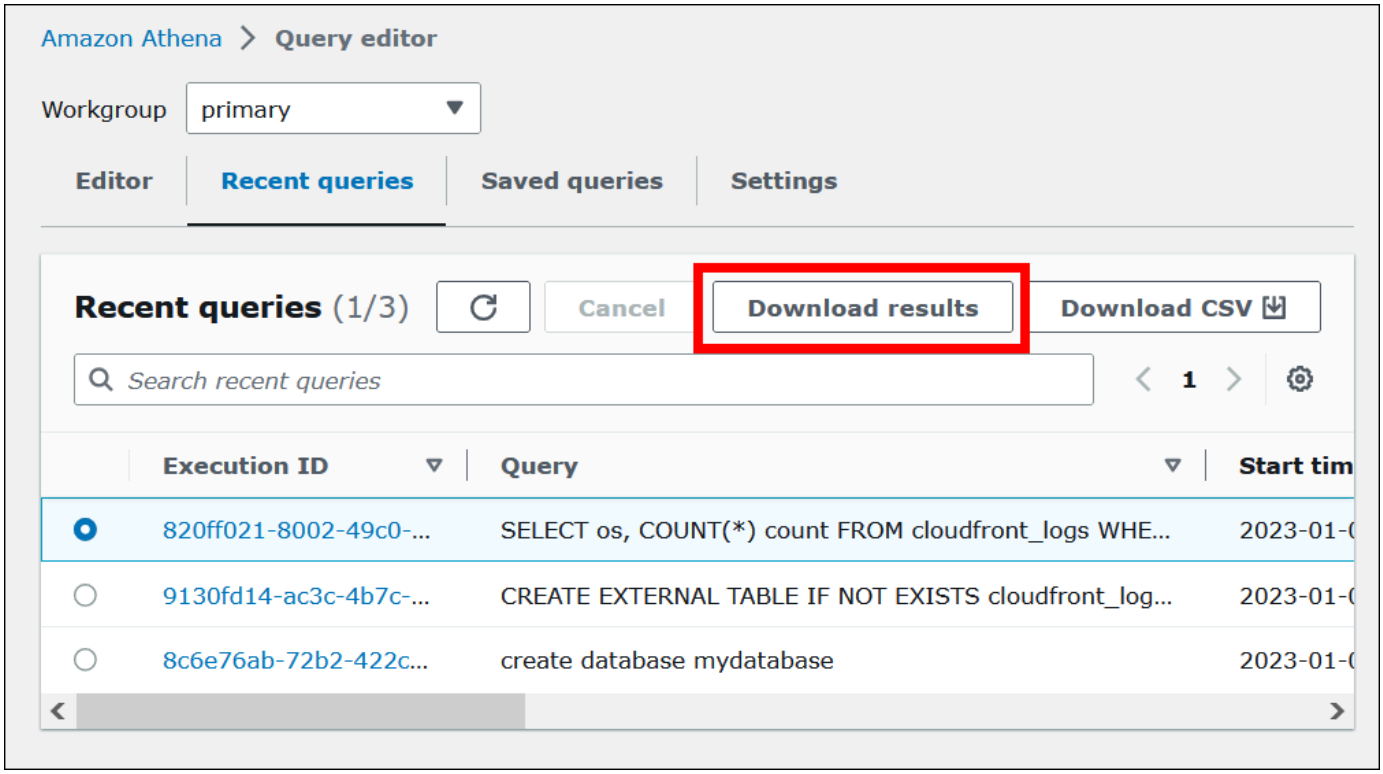


3. Pour enregistrer les résultats de la requête dans un fichier `.csv`, choisissez Download results (Télécharger les résultats).

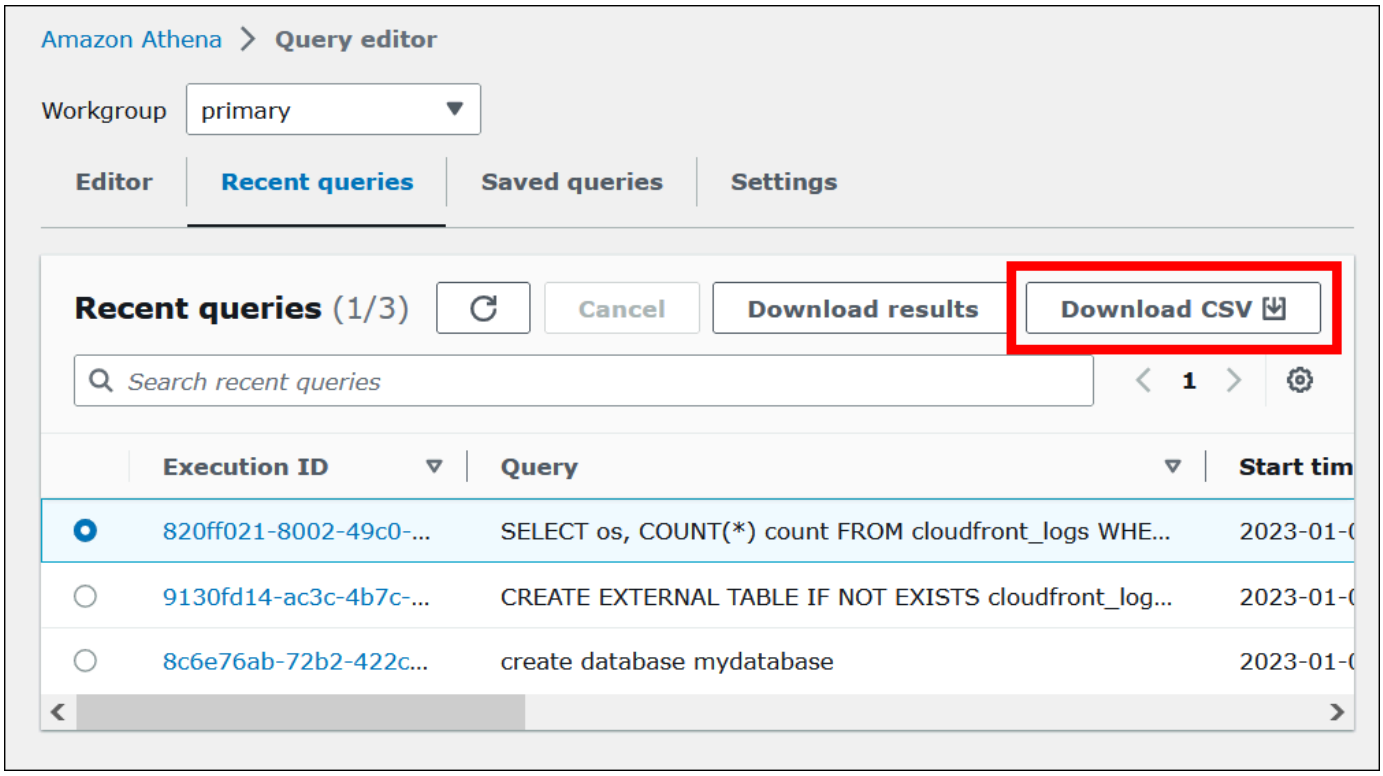
4. Pour afficher ou exécuter des requêtes précédentes, choisissez l'onglet Recent queries (Requêtes récentes).



5. Pour télécharger les résultats d'une requête précédente depuis l'onglet Recent queries (Requêtes récentes), sélectionnez la requête, puis choisissez Download results (Télécharger les résultats). Les requêtes sont conservées pendant 45 jours.



6. Pour télécharger une ou plusieurs chaînes de requêtes SQL récentes dans un fichier CSV, choisissez Download CSV (Télécharger CSV).



Pour plus d'informations, consultez Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie.

Enregistrement de vos requêtes

Vous pouvez enregistrer les requêtes que vous créez ou modifier dans l'éditeur de requêtes avec un nom. Athena stocke ces requêtes sur l'onglet Saved queries (Requêtes enregistrées). Vous pouvez utiliser l'onglet Saved queries (Requêtes enregistrées) pour rappeler, exécuter, renommer ou supprimer vos requêtes enregistrées. Pour plus d'informations, consultez Utilisation de requêtes enregistrées.

Raccourcis clavier et suggestions de saisie anticipée

L'éditeur de requêtes Athena fournit de nombreux raccourcis clavier pour des actions telles que l'exécution d'une requête, le formatage d'une requête, les opérations de ligne ainsi que la recherche et le remplacement. Pour plus d'informations et une liste complète des raccourcis, consultez Améliorer la productivité en utilisant les raccourcis clavier dans l'éditeur de requêtes Amazon Athena sur le blog AWS Big Data.

L'éditeur de requêtes Athena prend en charge les suggestions de code de saisie anticipée pour une expérience de création de requêtes plus rapide. Pour vous aider à écrire des requêtes SQL avec une précision et une efficacité accrues, il propose les fonctionnalités suivantes :

- Au fur et à mesure que vous tapez, des suggestions apparaissent en temps réel pour les mots-clés, les variables locales, les extraits et les éléments du catalogue.
- Lorsque vous tapez le nom d'une base de données ou d'une table suivi d'un point, l'éditeur affiche facilement une liste de tables ou de colonnes parmi lesquelles choisir.
- Lorsque vous passez le pointeur sur une suggestion d'extrait, un résumé présente un bref aperçu de la syntaxe et de l'utilisation de l'extrait.
- Pour améliorer la lisibilité du code, les mots-clés et leurs règles de mise en surbrillance ont également été mis à jour pour s'aligner sur la dernière syntaxe de Trino et Hive.

Cette caractéristique est activée par défaut. Pour activer ou désactiver cette fonctionnalité, utilisez les préférences de l'éditeur de code (icône en forme de roue dentée) en bas à droite de la fenêtre de l'éditeur de requêtes.

Connexion à d'autres sources de données

Ce tutoriel a utilisé une source de données dans Simple Storage Service (Amazon S3) au format CSV. Pour plus d'informations sur l'utilisation d'Athena avec AWS Glue, consultez. Utilisation AWS Glue pour se connecter à des sources de données dans Amazon S3 Vous pouvez également connecter Athena à diverses sources de données en utilisant des pilotes ODBC et JDBC, des métastores Hive externes et des connecteurs de sources de données Athena. Pour plus d'informations, voir Connexion aux sources de données.

Connexion aux sources de données

Vous pouvez utiliser Amazon Athena pour interroger des données stockées dans différents emplacements et différents formats dans un jeu de données. Ce jeu de données peut être au format CSV, JSON, Avro, Parquet ou dans un autre format.

Les tables et bases de données que vous utilisez dans Athena pour exécuter des requêtes sont basées sur des métadonnées. Les métadonnées sont des données relatives aux données sous-jacentes de votre jeu de données. La façon dont ces métadonnées décrivent votre jeu de données est appelée schéma. Par exemple, un nom de table, les noms de colonne de la table et le type de données de chaque colonne sont tous des schémas, enregistrés en tant que métadonnées, qui

décrivent un jeu de données sous-jacent. Dans Athena, un système permettant d'organiser les métadonnées est un catalogue de données ou un métastore. La combinaison d'un jeu de données et du catalogue de données qui le décrit est appelée source de données.

La relation entre les métadonnées et un jeu de données sous-jacent dépend du type de source de données que vous utilisez. Les sources de données relationnelles telles que MySQL, PostgreSQL et SQL Server intègrent étroitement les métadonnées au jeu de données. Dans ces systèmes, les métadonnées sont le plus souvent écrites au même moment que les données. D'autres sources de données, telles que celles créées à l'aide de Hive, vous permettent de définir des métadonnées on-the-fly lorsque vous lisez le jeu de données. Le jeu de données peut se présenter sous différents formats, par exemple CSV, JSON, Parquet ou Avro.

Athéna soutient nativement le. AWS Glue Data Catalog AWS Glue Data Catalog Il s'agit d'un catalogue de données construit sur d'autres ensembles de données et sources de données tels qu'Amazon S3, Amazon Redshift et Amazon DynamoDB. Vous pouvez également connecter Athena à d'autres sources de données à l'aide de divers connecteurs.

Rubriques

Intégration avec AWS Glue

AWS Glueest un ETL entièrement géré (extraction, transformation et chargement) Service AWS. L'une de ses capacités clés est d'analyser et de classer les données. Vous pouvez utiliser des AWS Glue robots d'exploration pour déduire automatiquement le schéma de base de données et de table à partir de vos données dans Amazon S3 et stocker les métadonnées associées dans le. AWS Glue Data Catalog

Athena utilise le AWS Glue Data Catalog pour stocker et récupérer les métadonnées de table pour les données Amazon S3 dans votre compte Amazon Web Services. Les métadonnées de la table permettent au moteur de requête Athena de savoir comment trouver, lire et traiter les données que vous souhaitez interroger.

Pour créer une base de données et un schéma de table dans le AWS Glue Data Catalog, vous pouvez exécuter un AWS Glue robot d'exploration depuis Athena sur une source de données, ou vous pouvez exécuter des requêtes DDL (Data Definition Language) directement dans l'éditeur de requêtes Athena. Ensuite, à l'aide de la base de données et du schéma de table que vous avez créés, vous pouvez utiliser des requêtes DML (Data Manipulation Language) dans Athena pour interroger les données.

Vous pouvez enregistrer et AWS Glue Data Catalog utiliser un compte autre que le vôtre. Après avoir configuré les autorisations IAM requises pour AWS Glue, vous pouvez utiliser Athena pour exécuter des requêtes entre comptes. Pour plus d'informations, consultez Accès entre comptes aux catalogues de données AWS Glue.

Pour plus d'informations sur le AWS Glue Data Catalog, consultez la section Catalogue de données et robots d'exploration AWS Glue dans le Guide du AWS Glue développeur.

Pour un article illustratif expliquant comment utiliser AWS Glue et comment traiter les données XML par Athena, consultez la section Traitement et analyse de fichiers XML volumineux et très imbriqués à l'aide d' AWS Glue Amazon Athena sur le blog Big Data. AWS

Des frais distincts s'appliquent à AWS Glue. Pour en savoir plus, consultez AWS Glue Tarification.

Rubriques

- Utilisation AWS Glue pour se connecter à des sources de données dans Amazon S3
- Enregistrer un compte AWS Glue Data Catalog depuis un autre compte
- Bonnes pratiques lors de l'utilisation d'Athena avec AWS Glue
- Utilisation du AWS CLI pour recréer une AWS Glue base de données et ses tables

Utilisation AWS Glue pour se connecter à des sources de données dans Amazon S3

Athena peut se connecter à vos données stockées dans Amazon S3 en utilisant le AWS Glue Data Catalog pour stocker des métadonnées telles que les noms de tables et de colonnes. Une fois la connexion établie, vos bases de données, tables et vues apparaissent dans l'éditeur de requête d'Athena.

Pour définir les informations de schéma AWS Glue à utiliser, vous pouvez créer un AWS Glue robot de recherche pour récupérer les informations automatiquement, ou vous pouvez ajouter manuellement une table et saisir les informations du schéma.

Création d'un AWS Glue crawler

Vous pouvez créer un crawler en commençant dans la console Athena, puis en utilisant la console AWS Glue de manière intégrée. Lorsque vous créez le crawler, vous spécifiez un emplacement de données à analyser dans Simple Storage Service (Amazon S3).

Pour créer un robot d'exploration à AWS Glue partir de la console Athena

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans l'éditeur de requêtes, à côté de Tables and views (Tables et vues), choisissez Create (Créer) puis choisissez le crawler AWS Glue .
3. Sur la page Add crawler (Ajouter un crawler) de la console AWS Glue, procédez comme suit pour créer un crawler. Pour plus d'informations, consultez les sections Utilisation AWS Glue des robots d'exploration dans ce guide et Remplissage du AWS Glue Data Catalog manuel du AWS Glue développeur.

> **Note**
>
> Athena ne reconnaît pas les modèles d'exclusion que vous spécifiez pour un AWS Glue robot d'exploration. Par exemple, si vous disposez d'un compartiment Simple Storage Service (Amazon S3) contenant à la fois des fichiers `.csv` et `.json` et que vous excluez les fichiers `.json` du Crawler, Athena interroge les deux groupes de fichiers. Pour éviter cela, placez les fichiers que vous voulez exclure dans un autre emplacement.

Ajout d'une table à l'aide d'un formulaire

La procédure suivante vous montre comment utiliser la console Athena pour ajouter une table à l'aide du formulaire Create Table From S3 bucket data (Créer une table à partir des données du compartiment S3).

Ajout d'une table et saisie des informations de schéma à l'aide d'un formulaire

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans l'éditeur de requêtes, à côté de Tables and views (Tables et vues), choisissez Create (Créer) puis choisissez S3 bucket data (Données de compartiment S3).
3. Dans le formulaire Create Table From S3 bucket data (Créer une table à partir des données du compartiment S3), pour Table name (Nom de la table), saisissez le nom de la table.

4. Pour Database configuration (Configuration de la base de données), choisissez une base de données existante ou créez-en une.
5. Pour Location of Input Data Set (Emplacement du jeu de données en entrée), spécifiez le chemin d'accès dans Simple Storage Service (Amazon S3) au dossier contenant le jeu de données que vous souhaitez traiter. N'incluez pas de nom de fichier dans le chemin. Athena analyse tous les fichiers dans le dossier que vous spécifiez. Si vos données sont déjà partitionnées (par exemple,

   s3://DOC-EXAMPLE-BUCKET/logs/year=2004/month=12/day=11/), saisissez uniquement le chemin de base (par exemple, s3://DOC-EXAMPLE-BUCKET/logs/).
6. Pour Data Format (Format des données), choisissez l'une des options suivantes :

   - Pour le type de table, choisissez Apache Hive, Apache Iceberg ou Delta Lake. Athena utilise le type de table Apache Hive par défaut. Pour plus d'informations sur l'interrogation des tables Apache Iceberg dans Athena, voir Utilisation des tables Apache Iceberg. Pour plus d'informations sur l'utilisation des tables Delta Lake dans Athena, voir Interrogation des tables Linux Foundation Delta Lake.
   - Pour File format (Format de fichier), choisissez le format de fichier ou de journal dans lequel se trouvent vos données.
     - Pour l'option Text File with Custom Delimiters (Fichier texte avec délimiteurs personnalisés), spécifiez un Field Terminator (Délimiteur de champ) (c'est-à-dire un délimiteur de colonne). Vous pouvez éventuellement spécifier un délimiteur de collection qui marque la fin d'un type de tableau ou un délimiteur de collection qui marque la fin d'un type de données cartographiques.
   - SerDe bibliothèque — Une bibliothèque SerDe (sérialiseur-désérialiseur) analyse un format de données particulier afin qu'Athena puisse créer une table pour celui-ci. Pour la plupart des formats, une SerDe bibliothèque par défaut est choisie pour vous. Pour les formats suivants, choisissez une bibliothèque en fonction de vos exigences :
     - Apache Web Logs : choisissez la GrokSerDebibliothèque RegexSerDeou la bibliothèque. Pour RegexSerDe, fournissez une expression régulière dans la zone de définition de Regex. Pour GrokSerDe, fournissez une série d'expressions régulières nommées pour la `input.format` SerDe propriété. Les expressions régulières nommées sont plus faciles à lire et à gérer que les expressions régulières. Pour plus d'informations, consultez Interrogation des journaux Apache stockés dans Simple Storage Service (Amazon S3).
     - CSV — Choisissez LazySimpleSerDesi vos données séparées par des virgules ne contiennent pas de valeurs entre guillemets ou si elles utilisent le `java.sql.Timestamp` format. Choisissez SerDeOpenCSV si vos données incluent des guillemets ou utilisent

le format numérique UNIX `TIMESTAMP` pour (par exemple,). `1564610311` Pour plus d'informations, consultez LazySimpleSerDe pour les fichiers CSV, TSV et délimités de manière personnalisée et SerDe OpenCSV pour le traitement des fichiers CSV.

- JSON — Choisissez la bibliothèque JSON OpenX ou Hive. SerDe Les deux formats nécessitent que chaque document JSON soit sur une seule ligne de texte et que les champs ne soient pas séparés par des caractères de saut de ligne. L'OpenX SerDe offre quelques propriétés supplémentaires. Pour de plus amples informations sur ces propriétés, veuillez consulter OpenX JSON SerDe. Pour plus d'informations sur le Hive SerDe, consultezHive JSON SerDe.

Pour plus d'informations sur l'utilisation SerDe des bibliothèques dans Athena, consultez. SerDe et formats de données pris en charge

7. Pour les SerDe propriétés, ajoutez, modifiez ou supprimez des propriétés et des valeurs en fonction de la SerDe bibliothèque que vous utilisez et de vos besoins.

   - Pour ajouter une SerDe propriété, choisissez Ajouter une SerDe propriété.
   - Dans le champ Name (Nom), saisissez le nom de la propriété.
   - Dans le champ Value (Valeur), saisissez une valeur pour la propriété.
   - Pour supprimer une SerDe propriété, choisissez Supprimer.

8. Pour Table properties (Propriétés de table), choisissez ou modifiez les propriétés de la table en fonction de vos exigences.

   - Pour Write compression (Compression d'écriture), choisissez une option de compression. La disponibilité de l'option de compression d'écriture et des options de compression dépend du format des données. Pour plus d'informations, consultez Prise en charge de la compression Athena.
   - Pour Encryption (Chiffrement), sélectionnez Encrypted data set (Jeu de données chiffrées) si les données sous-jacentes sont chiffrées dans Amazon S3. Cette option définit la propriété de table `has_encrypted_data` sur « true » (vrai) dans l'instruction `CREATE TABLE`.

9. Pour Column details (Détails de colonne), saisissez les noms et les types de données des colonnes que vous souhaitez ajouter à la table.

   - Pour ajouter d'autres colonnes une par une, choisissez Add a column (Ajouter une colonne).
   - Pour ajouter rapidement d'autres colonnes, choisissez Bulk add columns (Ajouter des colonnes en bloc). Dans la zone de texte, saisissez une liste de colonnes séparées par

des virgules au format *column_name data_type*, *column_name data_type*[, ...], puis choisissez Add (Ajouter).

10. (Facultatif) Pour Partition details (Détails de partition), ajoutez un ou plusieurs noms de colonnes et des types de données. Le partitionnement permet de conserver les données associées en fonction des valeurs des colonnes et peut contribuer à réduire la quantité de données scannées par requête. Pour plus d'informations sur le partitionnement, voir Partitionnement de données dans Athena.
11. (Facultatif) Pour Bucketing (Mise en compartiments), vous pouvez spécifier une ou plusieurs colonnes contenant des lignes que vous souhaitez regrouper, puis placer ces lignes dans plusieurs compartiments. Cela vous permet d'interroger uniquement le compartiment que vous souhaitez lire lorsque la valeur des colonnes mises en compartiments est spécifiée.

    - Pour Buckets (Compartiments), sélectionnez une ou plusieurs colonnes comportant un grand nombre de valeurs uniques (par exemple, une clé primaire) et qui sont fréquemment utilisées pour filtrer les données de vos requêtes.
    - Pour Number of buckets (Nombre de compartiments), saisissez un nombre permettant aux fichiers d'avoir une taille optimale. Pour plus d'informations, veuillez consulter l'article 10 meilleurs conseils de réglage des performances pour Amazon Athena sur le blog AWS Big Data.
    - Pour spécifier vos colonnes mises en compartiments, l'instruction `CREATE TABLE` utilisera la syntaxe suivante :

    ```
    CLUSTERED BY (bucketed_columns) INTO number_of_buckets BUCKETS
    ```

    > **Note**
    >
    > L'option Bucketing (Mise en compartiments) n'est pas disponible pour le type de table Iceberg.

12. La zone Preview table query (Requête de prévisualisation de table) affiche l'instruction `CREATE TABLE` générée par les informations que vous avez saisies dans le formulaire. L'instruction de prévisualisation ne peut pas être modifiée directement. Pour modifier l'instruction, modifiez les champs du formulaire au-dessus de l'aperçu, ou créez directement l'instruction dans l'éditeur de requêtes au lieu d'utiliser le formulaire.

13. Choisissez Create table (Créer une table) pour exécuter l'instruction générée dans l'éditeur de requêtes et créer la table.

Enregistrer un compte AWS Glue Data Catalog depuis un autre compte

Vous pouvez utiliser la fonction de AWS Glue catalogue multi-comptes d'Athena pour enregistrer un AWS Glue catalogue à partir d'un compte autre que le vôtre. Après avoir configuré les autorisations IAM requises pour AWS Glue et enregistré le catalogue en tant que ressource `DataCatalog` Athena, vous pouvez utiliser Athena pour exécuter des requêtes inter-comptes. Pour plus d'informations sur la configuration des autorisations requises, voirAccès entre comptes aux catalogues de données AWS Glue.

La procédure suivante vous montre comment utiliser la console Athena pour configurer un AWS Glue Data Catalog dans un compte Amazon Web Services autre que le vôtre comme source de données.

Pour enregistrer un compte AWS Glue Data Catalog depuis un autre compte

1. Suivez les étapes de la rubrique Accès entre comptes aux catalogues de données AWS Glue pour vous assurer que vous avez les autorisations d'interroger le catalogue de données dans l'autre compte.
2. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
3. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

4. Choisissez Sources de données.

5. Dans le coin supérieur droit, choisissez Create data source (Créer une source de données).
6. Sur la page Choisir une source de données, pour Sources de données, choisissez S3 - AWS Glue Data Catalog, puis Next.
7. Sur la page Enter data source details (Saisir les détails de la source de données), dans la section AWS Glue Data Catalog, pour Choisir un AWS Glue Data Catalog, choisissez AWS Glue Data Catalog dans un autre compte.
8. Pour Data source details (Détails de la source de données), saisissez les informations suivantes :

   - Data source name (Nom de la source de données) – Saisissez le nom que vous souhaitez utiliser dans vos requêtes SQL pour faire référence au catalogue de données dans l'autre compte.
   - Description – (Facultative) Saisissez une description du catalogue de données dans l'autre compte.
   - ID du catalogue – Saisissez l'ID de compte Amazon Web Services à 12 chiffres du compte auquel appartient le catalogue de données. L'ID de compte Amazon Web Services est l'ID de catalogue.
9. (Facultatif) Pour Tags (Identifications), saisissez les paires clé-valeur que vous voulez associer à la source de données. Pour en savoir plus sur les identifications, consultez Étiquetage des ressources Athena.
10. Choisissez Suivant.
11. Sur la page Review and create (Vérifier et créer), vérifiez les informations que vous avez fournies, puis choisissez Create data source (Créer une source de données). La page Data source details (Détails de la source de données) répertorie les bases de données et les balises du catalogue de données que vous avez enregistré.
12. Choisissez Sources de données. Le catalogue de données que vous avez enregistré est répertorié dans la colonne Data source name (Nom de la source de données).
13. Pour afficher ou modifier les informations relatives au catalogue de données, choisissez le catalogue, puis choisissez Actions, Edit (Modifier).
14. Pour supprimer le nouveau catalogue de données, choisissez le catalogue, puis choisissez Actions, Supprimer.

Pour plus d'informations, consultez la section Interrogation de comptes AWS Glue Data Catalog multiples à l'aide d'Amazon Athena sur AWS le blog Big Data.

Bonnes pratiques lors de l'utilisation d'Athena avec AWS Glue

Lorsque vous utilisez Athéna avec le AWS Glue Data Catalog, vous pouvez l'utiliser AWS Glue pour créer des bases de données et des tables (schéma) à interroger dans Athéna, ou vous pouvez utiliser Athéna pour créer un schéma, puis les utiliser dans des services connexes. AWS Glue Cette rubrique fournit des considérations et les bonnes pratiques sur l'utilisation de l'une ou l'autre méthode.

En coulisses, Athena utilise Trino pour traiter les instructions DML et Hive pour traiter les instructions DDL qui créent et modifient le schéma. Avec ces technologies, il y a quelques conventions à suivre pour qu'Athéna et moi AWS Glue travaillons bien ensemble.

Dans cette rubrique

Noms de bases de données, de tables et de colonnes

Lorsque vous créez un schéma dans AWS Glue une requête dans Athéna, tenez compte des points suivants :

- Les caractères acceptables pour les noms de base de données, de tables et de colonnes AWS Glue doivent être une chaîne UTF-8 et doivent être en minuscules. Notez qu'Athena abaisse automatiquement les noms en majuscules dans les requêtes DDL lorsqu'elle crée des bases de données, des tables ou des colonnes. La longueur de la chaîne ne doit pas être inférieure à 1 ni supérieure à 255 octets. Les caractères pouvant être utilisés incluent des espaces.
- Actuellement, il est possible d'avoir des espaces au début des noms. Comme ces espaces de début peuvent être difficiles à détecter et peuvent entraîner des problèmes d'utilisabilité après leur création, évitez de créer par inadvertance des noms d'objets avec des espaces de début.
- Si vous utilisez un AWS::Glue::Database AWS CloudFormation modèle pour créer une AWS Glue base de données et que vous ne spécifiez pas de nom de base de données, génère AWS Glue automatiquement un nom de base de données au format `resource_name-random_string` incompatible avec Athena.
- Vous pouvez utiliser le gestionnaire de AWS Glue catalogue pour renommer les colonnes, mais pas les noms de tables ou de bases de données. Pour contourner cette limitation, vous devez utiliser une définition de l'ancienne base de données pour créer une base de données portant le nouveau nom. Vous utilisez ensuite les définitions des tables de l'ancienne base de données pour recréer les tables de la nouvelle base de données. Pour ce faire, vous pouvez utiliser le AWS Glue SDK AWS CLI ou. Pour les étapes, consultez Utilisation du AWS CLI pour recréer une AWS Glue base de données et ses tables.

Pour plus d'informations sur les bases de données et les tables dans AWS Glue, consultez la section Bases de données et tables du manuel du AWS Glue développeur.

Utiliser des AWS Glue crawlers

AWS Glue les robots d'exploration aident à découvrir le schéma des ensembles de données et à les enregistrer sous forme de tables dans le catalogue de AWS Glue données. Les Crawlers explorent vos données et en déterminent le schéma. De plus, le Crawler peut détecter et enregistrer des partitions. Pour plus d'informations, consultez Définition des Crawlers dans le Guide du développeur AWS Glue . Les tables de données dont l'analyse a abouti peuvent être interrogées à partir d'Athena.

> **Note**
>
> Athena ne reconnaît pas les modèles d'exclusion que vous spécifiez pour un AWS Glue robot d'exploration. Par exemple, si vous disposez d'un compartiment Simple Storage Service (Amazon S3) contenant à la fois des fichiers `.csv` et `.json` et que vous excluez les fichiers `.json` du Crawler, Athena interroge les deux groupes de fichiers. Pour éviter cela, placez les fichiers que vous voulez exclure dans un autre emplacement.

Planification d'un Crawler pour préserver la synchronisation entre le AWS Glue Data Catalog et Simple Storage Service (Amazon S3)

AWS Glue les robots d'exploration peuvent être configurés pour fonctionner selon un calendrier ou à la demande. Pour en savoir plus, consultez Planifications temporelles pour les tâches et les Crawlers dans le Guide du développeur AWS Glue .

Si des données arrivent pour une table partitionnée à une heure fixe, vous pouvez configurer un AWS Glue robot d'exploration pour qu'il s'exécute selon le calendrier prévu afin de détecter et de mettre à jour les partitions de table. Cela peut éliminer la nécessité d'exécuter une commande `MSCK REPAIR` potentiellement longue et coûteuse ou d'exécuter manuellement une commande `ALTER TABLE ADD PARTITION`. Pour en savoir plus, consultez Partitions de table dans le Guide du développeur AWS Glue .

Utilisation de plusieurs sources de données avec les Crawlers

Lorsqu'un AWS Glue robot d'exploration analyse Amazon S3 et détecte plusieurs répertoires, il utilise une heuristique pour déterminer où se trouve la racine d'une table dans la structure du répertoire et quels répertoires sont des partitions de la table. Dans certains cas, si le schéma détecté dans deux ou plusieurs répertoires est similaire, le Crawler peut les traiter comme des partitions et pas comme des tables distinctes. L'une des solutions pour aider le Crawler à découvrir les tables individuelles consiste à ajouter le répertoire racine de chaque table comme magasin de données du analyseur.

Les partitions suivantes dans Simple Storage Service (Amazon S3) constituent un example :

```
s3://DOC-EXAMPLE-BUCKET/folder1/table1/partition1/file.txt
s3://DOC-EXAMPLE-BUCKET/folder1/table1/partition2/file.txt
s3://DOC-EXAMPLE-BUCKET/folder1/table1/partition3/file.txt
s3://DOC-EXAMPLE-BUCKET/folder1/table2/partition4/file.txt
s3://DOC-EXAMPLE-BUCKET/folder1/table2/partition5/file.txt
```

Si les schémas de `table1` et de `table2` sont similaires, et qu'une seule source de données est définie sur `s3://DOC-EXAMPLE-BUCKET/folder1/` dans AWS Glue, le Crawler peut créer une seule table avec deux colonnes de partition : une colonne de partition qui contient `table1` et `table2`, et une deuxième colonne de partition qui contient les partitions de `partition1` à `partition5`.

Pour que le AWS Glue robot crée deux tables distinctes, configurez le robot de manière à ce qu'il dispose de deux sources de données`s3://DOC-EXAMPLE-BUCKET/folder1/table2`, `s3://DOC-EXAMPLE-BUCKET/folder1/table1/` et comme indiqué dans la procédure suivante.

Pour ajouter un magasin de données S3 à un robot d'exploration existant dans AWS Glue

1. Connectez-vous à la AWS Glue console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/glue/.
2. Dans le panneau de navigation, sélectionnez Crawlers. (Analyseurs)
3. Choisissez le lien vers votre crawler, puis choisissez Modification.
4. Étape 2 : Choisir des sources de données et des classificateurs, choisissez Modification.
5. Dans la section Sources de données, choisissez Ajouter une source de données.
6. Dans la boite de dialogue Ajouter une source de données pour le Chemin S3, choisissez Parcourir.
7. Choisissez le compartiment que vous souhaitez utiliser, ensuite choisissez Sélectionner un plan.

   La source de données que vous avez ajoutée apparaît dans la liste Sources de données.
8. Choisissez Suivant.
9. Sur la page Configurer les paramètres de sécurité, ou créez ou choisissez un rôle IAM pour le crawler, puis sélectionnez Suivant.
10. Assurez-vous que le chemin S3 se termine par une barre oblique, avant de choisir ensuite Ajouter une source de données S3.
11. Sur la page Régler la sortie et la planification, pour la Configuration de sortie, sélectionnez la base de données cible.
12. Choisissez Suivant.
13. Sur la page Vérifier et mettre à jour, passez en revue les choix que vous avez effectués. Pour modifier une étape, sélectionnez Modification.
14. Choisissez Mettre à jour.

Synchronisation du schéma de la partition pour éviter « HIVE_PARTITION_SCHEMA_MISMATCH »

Pour chaque table du catalogue de AWS Glue données comportant des colonnes de partition, le schéma est stocké au niveau de la table et pour chaque partition individuelle de la table. Le schéma des partitions est renseigné par un AWS Glue robot d'exploration en fonction de l'échantillon de données qu'il lit dans la partition. Pour plus d'informations, consultez Utilisation de plusieurs sources de données avec les Crawlers.

Lorsqu'Athena exécute une requête, il valide le schéma de la table et le schéma de toute partition nécessaire à la requête. La validation compare les types de données des colonnes dans l'ordre et s'assure qu'ils correspondent pour les colonnes qui se chevauchent. Ceci empêche les opérations inattendues, telles que l'ajout ou la suppression de colonnes depuis le milieu d'une table. Si Athena détecte que le schéma d'une partition diffère du schéma de la table, Athena risque de ne pas pouvoir traiter la requête et échoue avec `HIVE_PARTITION_SCHEMA_MISMATCH`.

Il existe deux moyens de corriger ce problème. Tout d'abord, si les données ont été ajoutées accidentellement, vous pouvez supprimer les fichiers de données à l'origine de la différence de schéma, supprimer la partition et exécuter à nouveau le robot d'indexation sur les données. Deuxièmement, vous pouvez supprimer la partition individuelle, puis exécuter `MSCK REPAIR` dans Athena pour recréer la partition à l'aide du schéma de la table. Cette deuxième option fonctionne uniquement si vous êtes certain que le schéma appliqué continuera de lire les données correctement.

Mise à jour des métadonnées de table

Après un crawl, le AWS Glue robot attribue automatiquement certaines métadonnées aux tables afin de les rendre compatibles avec d'autres technologies externes telles qu'Apache Hive, Presto et Spark. De temps en temps, le Crawler peut attribuer de manière incorrecte les propriétés des métadonnées. Corrigez manuellement les propriétés AWS Glue avant d'interroger la table à l'aide d'Athena. Pour en savoir plus, consultez Affichage et modification des détails de table dans le Guide du développeur AWS Glue .

AWS Glue peut mal affecter les métadonnées lorsqu'un fichier CSV contient des guillemets autour de chaque champ de données, ce qui entraîne une erreur de `serializationLib` propriété. Pour plus d'informations, consultez Données CSV entre guillemets.

Utilisation de fichiers CSV

Les fichiers CSV ont occasionnellement des guillemets autour des valeurs prévues pour chaque colonne, et il peut y avoir des valeurs d'en-tête incluses dans les fichiers CSV qui ne font pas partie

des données à analyser. Lorsque vous créez un schéma AWS Glue à partir de ces fichiers, suivez les instructions de cette section.

Données CSV entre guillemets

Vous pouvez avoir un fichier CSV dont les champs de données sont entre guillemets, comme dans l'exemple suivant :

```
"John","Doe","123-555-1231","John said \"hello\""
"Jane","Doe","123-555-9876","Jane said \"hello\""
```

Pour exécuter une requête dans Athena sur une table créée à partir d'un fichier CSV contenant des valeurs entre guillemets, vous devez modifier les propriétés de la table AWS Glue pour utiliser l'OpenCSV. SerDe Pour plus d'informations sur SerDe OpenCSV, consultez. SerDe OpenCSV pour le traitement des fichiers CSV

Pour modifier les propriétés du tableau dans la AWS Glue console

1. Dans le volet de navigation de la AWS Glue console, sélectionnez Tables.
2. Choisissez le lien de la table que vous souhaitez modifier, puis sélectionnez Action, Modifier la table.
3. Sur la page Modifier le tableau, procédez au modifications suivantes :
   - Dans la Bibliothèque de sérialisation, saisissez `org.apache.hadoop.hive.serde2.OpenCSVSerde`.
   - Pour Serde parameters (Paramètres Serde), saisissez les valeurs suivantes pour les clés `escapeChar`, `quoteChar` et `separatorChar` :
     - Pour `escapeChar`, saisissez une barre oblique inverse (****).
     - Pour `quoteChar`, saisissez un guillemet double (**"**).
     - Pour `separatorChar`, saisissez une virgule (**,**).
4. Choisissez Enregistrer.

Pour en savoir plus, consultez Affichage et modification des détails de table dans le Guide du développeur AWS Glue .

Mettre à jour les propriétés d'une AWS Glue table par programmation

Vous pouvez utiliser l'opération AWS Glue UpdateTableAPI ou la commande update-table CLI pour modifier le `SerDeInfo` bloc dans la définition de la table, comme dans l'exemple JSON suivant.

```
"SerDeInfo": {
   "name": "",
   "serializationLib": "org.apache.hadoop.hive.serde2.OpenCSVSerde",
   "parameters": {
      "separatorChar": ","
      "quoteChar": "\""
      "escapeChar": "\\"
      }
},
```

Fichiers CSV avec en-têtes

Lorsque vous définissez une table dans Athena avec une instruction `CREATE TABLE`, vous pouvez utiliser la propriété de table `skip.header.line.count` pour ignorer les en-têtes dans vos données CSV, comme dans l'exemple suivant.

```
...
STORED AS TEXTFILE
LOCATION 's3://DOC-EXAMPLE-BUCKET/csvdata_folder/';
TBLPROPERTIES ("skip.header.line.count"="1")
```

Vous pouvez également supprimer les en-têtes CSV au préalable afin que les informations d'en-tête ne soient pas incluses dans les résultats des requêtes Athena. L'un des moyens d'y parvenir consiste à utiliser AWS Glue des tâches qui exécutent des tâches d'extraction, de transformation et de chargement (ETL). Vous pouvez écrire des scripts en AWS Glue utilisant un langage qui est une extension du dialecte PySpark Python. Pour plus d'informations, consultez la section Création de tâches dans AWS Glue dans le manuel du AWS Glue développeur.

L'exemple suivant montre une fonction dans un AWS Glue script qui écrit un cadre dynamique en utilisant l'`from_options`option de `writeHeader` format et lui attribuant la valeur false, ce qui supprime les informations d'en-tête :

```
glueContext.write_dynamic_frame.from_options(frame = applymapping1, connection_type =
 "s3", connection_options = {"path": "s3://DOC-EXAMPLE-BUCKET/MYTABLEDATA/"}, format =
 "csv", format_options = {"writeHeader": False}, transformation_ctx = "datasink2")
```

AWS Glue indexation et filtrage des partitions

Lors de l'interrogation de tables partitionnées, Athena récupère et filtre les partitions de table disponibles vers le sous-ensemble correspondant à votre requête. À mesure que de nouvelles données et partitions sont ajoutées, il faut plus de temps pour traiter les partitions et l'exécution des requêtes peut augmenter. Si vous disposez d'une table avec un grand nombre de partitions qui croît au fil du temps, envisagez d'utiliser indexation et filtrage des partitions AWS Glue . L'indexation des partitions permet à Athena d'optimiser le traitement des partitions et d'améliorer les performances des requêtes sur les tables fortement partitionnées. La configuration du filtrage de partition dans les propriétés d'une table s'effectue en deux étapes :

1. Création d'un index de partition dans AWS Glue.
2. Activation du filtrage des partitions pour la table.

Création d'un index de partition

Pour connaître les étapes de création d'un index de partition dans AWS Glue, consultez la section Utilisation des index de partition dans le Guide du AWS Glue développeur. Pour connaître les limites relatives aux index de partition dans AWS Glue, consultez la section À propos des index de partition de cette page.

Activation du filtrage de partition

Pour activer le filtrage des partitions pour la table, vous devez définir une nouvelle propriété de table dans AWS Glue. Pour savoir comment définir les propriétés d'une table dans AWS Glue, reportez-vous à la page Configuration de la projection par partition. Lorsque vous modifiez les détails de la table dans AWS Glue, ajoutez la paire clé-valeur suivante à la section Propriétés de la table :

- Pour Key (Clé), ajoutez `partition_filtering.enabled`
- Pour Value (Valeur), ajoutez `true`

Vous pouvez désactiver le filtrage de partition sur cette table à tout moment en définissant la valeur `partition_filtering.enabled` sur `false`.

Une fois que vous avez terminé les étapes ci-dessus, vous pouvez revenir à la console Athena pour interroger les données.

Pour plus d'informations sur l'utilisation de l'indexation et du filtrage des partitions, consultez la section Améliorer les performances des requêtes Amazon Athena à AWS Glue Data Catalog l'aide d'index de partition sur AWS le blog Big Data.

Utilisation de données géospatiales

AWS Glue ne prend pas en charge nativement le texte connu (WKT), le binaire connu (WKB) ou les autres types de données PostGIS. Le AWS Glue classificateur analyse les données géospatiales et les classe à l'aide des types de données pris en charge pour le format, par exemple pour CSV. `varchar` Comme pour les autres AWS Glue tables, vous devrez peut-être mettre à jour les propriétés des tables créées à partir de données géospatiales pour permettre à Athena d'analyser ces types de données tels quels. Pour plus d'informations, consultez Utiliser des AWS Glue crawlers et Utilisation de fichiers CSV. Athena ne sera peut-être pas en mesure d'analyser certains types de données géospatiales dans les tables telles quelles. AWS Glue Pour de plus amples informations sur l'utilisation des données géospatiales dans Athena, veuillez consulter Interrogation de données géospatiales.

Utiliser des AWS Glue jobs pour l'ETL avec Athena

AWS Glue les jobs exécutent des opérations ETL. Une AWS Glue tâche exécute un script qui extrait les données des sources, les transforme et les charge dans des cibles. Pour plus d'informations, consultez la section Création de tâches dans AWS Glue dans le manuel du AWS Glue développeur.

Création de tables à l'aide d'Athena pour les tâches ETL AWS Glue

Les tables que vous créez dans Athena doivent se voir ajouter une propriété de table appelée `classification`, qui identifie le format des données. Cela permet d' AWS Glue utiliser les tables pour les tâches ETL. Les valeurs de classification peuvent être `avro`, `csv`, `json`, `orc`, `parquet`, ou `xml`. Voici un exemple d'instruction `CREATE TABLE` dans Athena :

```
CREATE EXTERNAL TABLE sampleTable (
  column1 INT,
  column2 INT
  ) STORED AS PARQUET
  TBLPROPERTIES (
  'classification'='parquet')
```

Si la propriété de table n'a pas été ajoutée lors de la création de la table, vous pouvez l'ajouter à l'aide de la AWS Glue console.

Pour ajouter la propriété de la table de classification à l'aide de la AWS Glue console

1. Connectez-vous à la AWS Glue console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/glue/.
2. Dans le panneau de navigation de la console, choisissez Tableaux.
3. Choisissez le lien de la table que vous souhaitez modifier, puis sélectionnez Action, Modifier la table.
4. Faites défiler vers le bas jusqu'à la section des Propriétés du tableau.
5. Choisissez Ajouter.
6. Pour Key (Clé), saisissez **`classification`**.
7. Pour la Valeur, entrez un type de données (**`json`** par exemple).
8. Choisissez Enregistrer.

   Dans la section Détails de la table, le type de données que vous avez saisi apparaîssent dans le champ Classification du tableau.

Pour en savoir plus, consultez Utilisation des tables dans le Guide du développeur AWS Glue .

Utilisation des tâches ETL pour optimiser les performances des requêtes

AWS Glue les jobs peuvent vous aider à transformer les données dans un format qui optimise les performances des requêtes dans Athena. Les formats de données ont un grand impact sur les performances et les coûts d'interrogation dans Athena.

Nous recommandons d'utiliser les formats de données Parquet et ORC. AWS Glue prend en charge l'écriture dans ces deux formats de données, ce qui peut vous permettre de transformer plus facilement et plus rapidement les données dans un format optimal pour Athena. Pour plus d'informations sur ces formats et sur d'autres moyens d'améliorer les performances, consultez les 10 meilleurs conseils d'optimisation des performances pour Amazon Athena.

Conversion des types de données SMALLINT et TINYINT en INT lors de la conversion en ORC

Pour réduire le risque qu'Athena ne soit pas en mesure de lire les types de `TINYINT` données `SMALLINT` et de données produits par une tâche AWS Glue ETL, `SMALLINT` `TINYINT` convertissez-les `INT` en utilisant l'assistant ou en écrivant un script pour une tâche ETL.

Automatiser les AWS Glue tâches pour l'ETL

Vous pouvez configurer les tâches AWS Glue ETL pour qu'elles s'exécutent automatiquement en fonction de déclencheurs. Cette fonctionnalité est idéale lorsque des données provenant de l'extérieur AWS sont transférées vers un compartiment Amazon S3 dans un format sous-optimal pour les requêtes dans Athena. Pour plus d'informations, consultez la rubrique Déclenchement des tâches AWS Glue dans le Guide du développeur AWS Glue .

Utilisation du AWS CLI pour recréer une AWS Glue base de données et ses tables

Il n'est pas possible de renommer directement une AWS Glue base de données, mais vous pouvez copier sa définition, modifier la définition et utiliser la définition pour recréer la base de données sous un autre nom. De même, vous pouvez copier les définitions des tables de l'ancienne base de données, modifier les définitions et utiliser les définitions modifiées pour recréer les tables de la nouvelle base de données.

> **Note**
>
> La méthode présentée ne copie pas le partitionnement des tables.

La procédure suivante pour Windows suppose que vous êtes AWS CLI configuré pour la sortie JSON. Pour modifier le format de sortie par défaut dans le AWS CLI, exécutez`aws configure`.

Pour copier une AWS Glue base de données à l'aide du AWS CLI

1. À l'invite de commandes, exécutez la AWS CLI commande suivante pour récupérer la définition de la AWS Glue base de données que vous souhaitez copier.

   ```
   aws glue get-database --name database_name
   ```

   Pour obtenir plus d'informations sur la commande `get-database`, consultez get-database.
2. Enregistrez la sortie JSON dans un fichier portant le nom de la nouvelle base de données (par exemple,*new_database_name*`.json`) sur votre bureau.
3. Ouvrez le fichier *new_database_name*`.json` dans un éditeur de texte.
4. Dans le fichier JSON, effectuez les opérations suivantes :

   a. Retirez l'`{ "Database":`entrée extérieure et le support de fermeture correspondant `}` à la fin du fichier.

b. Remplacez l'`Name`entrée par le nouveau nom de base de données.

c. Supprimez le champ `CatalogId`.

5. Enregistrez le fichier.

6. À l'invite de commandes, exécutez la AWS CLI commande suivante pour utiliser le fichier de définition de base de données modifié afin de créer la base de données sous le nouveau nom.

```
aws glue create-database --database-input "file://~/Desktop\new_database_name.json"
```

Pour obtenir plus d'informations sur la commande `create-database`, consultez create-database. Pour plus d'informations sur le chargement de AWS CLI paramètres depuis un fichier, consultez la section Chargement de AWS CLI paramètres depuis un fichier dans le Guide de AWS Command Line Interface l'utilisateur.

7. Pour vérifier que la nouvelle base de données a été créée dans AWS Glue, exécutez la commande suivante :

```
aws glue get-database --name new_database_name
```

Vous êtes maintenant prêt à obtenir la définition d'une table à copier dans la nouvelle base de données, à modifier la définition et à utiliser la définition modifiée pour recréer la table dans la nouvelle base de données. Cette procédure ne modifie pas le nom de la table.

Pour copier un AWS Glue tableau à l'aide du AWS CLI

1. À l'invite de commande, exécutez la AWS CLI commande suivante.

```
aws glue get-table --database-name database_name --name table_name
```

Pour obtenir plus d'informations sur la commande `get-table`, consultez get-table.

2. Enregistrez la sortie JSON dans un fichier portant le nom de la table (par exemple, *`table_name`*.json) sur votre bureau Windows.

3. Ouvrez le fichier dans un éditeur de texte.

4. Dans le fichier JSON, supprimez l'entrée `{"Table":` extérieure et le crochet de fermeture correspondant `}` à la fin du fichier.

5. Dans le fichier JSON, supprimez les entrées suivantes et leurs valeurs :

- `DatabaseName` – Cette entrée n'est pas obligatoire, car la commande CLI `create-table` utilise le paramètre `--database-name`.
- `CreateTime`
- `UpdateTime`
- `CreatedBy`
- `IsRegisteredWithLakeFormation`
- `CatalogId`
- `VersionId`

6. Enregistrez le fichier de définition de table.
7. À l'invite de commande, exécutez la AWS CLI commande suivante pour recréer la table dans la nouvelle base de données :

```
aws glue create-table --database-name new_database_name --table-input "file://~/
Desktop\table_name.json"
```

Pour obtenir plus d'informations sur la commande `create-table`, consultez create-table.

La table apparaît désormais dans la nouvelle base de données AWS Glue et peut être interrogée auprès d'Athena.

8. Répétez les étapes pour copier chaque table supplémentaire dans la nouvelle base de données dans AWS Glue.

Utilisation du connecteur de données Athena pour un métastore Hive externe

Vous pouvez utiliser le connecteur de données Amazon Athena pour le métastore Hive externe afin d'interroger les jeux de données Simple Storage Service (Amazon S3) qui utilisent un métastore Apache Hive. Aucune migration des métadonnées vers le AWS Glue Data Catalog n'est nécessaire. Dans la console de gestion Athena, vous configurez une fonction Lambda pour communiquer avec le métastore Hive qui se trouve dans votre VPC privé, puis vous la connectez au métastore. La connexion de Lambda à votre métastore Hive est sécurisée par un canal privé Amazon VPC et n'utilise pas l'internet public. Vous pouvez fournir votre propre code de fonction Lambda ou utiliser l'implémentation par défaut du connecteur de données Athena pour le métastore Hive externe.

Rubriques

Présentation des fonctions

Avec le connecteur de données Athena pour métastore Hive externe, vous pouvez effectuer les tâches suivantes :

- Utiliser la console Athena pour enregistrer des catalogues personnalisés et exécuter des requêtes à l'aide de ceux-ci.
- Définir des fonctions Lambda pour différents métastores externes de Hive et les joindre dans des requêtes Athena.
- Utilisez les métastores Hive AWS Glue Data Catalog et vos métastores externes dans la même requête Athena.
- Spécifiez un catalogue dans le contexte d'exécution de requête en tant que catalogue par défaut actuel. Cela supprime la nécessité de préfixer les noms de catalogue aux noms de base de données dans vos requêtes. Au lieu d'utiliser la syntaxe `catalog.database.table`, vous pouvez utiliser `database.table`.
- Utilisez une variété d'outils pour exécuter des requêtes qui font référence aux métastores Hive externes. Vous pouvez utiliser la console Athena, le AWS SDK AWS CLI, les API Athena et les pilotes JDBC et ODBC Athena mis à jour. Les pilotes mis à jour prennent en charge les catalogues personnalisés.

Prise en charge de l'API

Le connecteur de données Athena pour le métastore Hive externe comprend la prise en charge des opérations d'API d'enregistrement du catalogue et des opérations de l'API de métadonnées.

- Enregistrement de catalogue – Enregistrez les catalogues personnalisés pour les métastores Hive externes et les sources de données fédérées.
- Métadonnées : utilisez les API de métadonnées pour fournir des informations de base de données AWS Glue et de table pour tout catalogue que vous enregistrez auprès d'Athena.
- Client JAVA SDK Athena – Utilisez les API d'enregistrement de catalogue, les API de métadonnées et la prise en charge des catalogues dans l'opération `StartQueryExecution` dans le client Java SDK Athena mis à jour.

Implémentation de référence

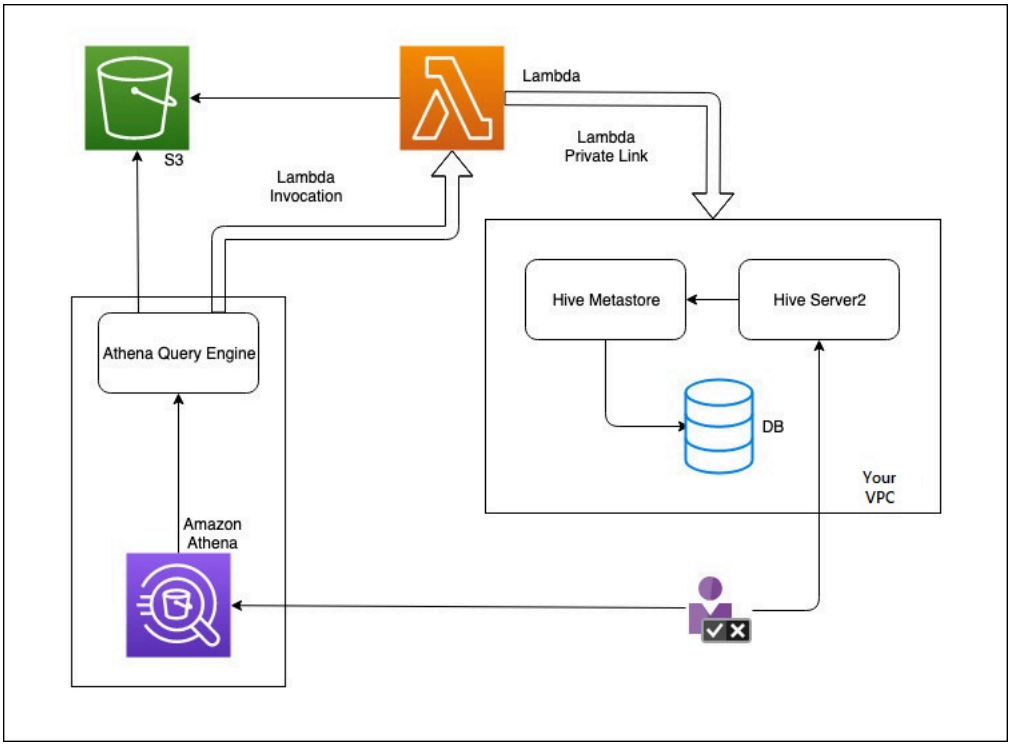
Athena fournit une implémentation de référence pour la fonction Lambda qui se connecte aux métastores Hive externes. L'implémentation de référence est fournie sous GitHub forme de projet open source sur le metastore Athena Hive.

L'implémentation de référence est disponible sous la forme AWS SAM des deux applications suivantes dans le AWS Serverless Application Repository (SAR). Vous pouvez utiliser l'une de ces applications dans le SAR pour créer vos propres fonctions Lambda.

- **`AthenaHiveMetastoreFunction`** – Fichier `.jar` de fonction Lambda uber. Un JAR « uber » (également connu sous le nom de gros JAR ou JAR avec dépendances) est un fichier `.jar` qui contient à la fois un programme Java et ses dépendances dans un seul fichier.
- **`AthenaHiveMetastoreFunctionWithLayer`** – Couche Lambda et fichier `.jar` de fonction Lambda mince.

Flux de travail

Le diagramme suivant montre comment Athena interagit avec votre métastore Hive externe.



Dans ce flux de travail, votre métastore Hive connecté à une base de données se trouve à l'intérieur de votre VPC. Vous utilisez Hive Server2 pour gérer votre métastore Hive à l'aide de la CLI Hive.

Le flux de travail permettant d'utiliser des métastores Hive externes depuis Athena inclut les étapes suivantes.

1. Vous créez une fonction Lambda qui connecte Athena au métastore Hive se trouvant à l'intérieur de votre VPC.
2. Vous enregistrez un nom de catalogue unique pour votre métastore Hive et un nom de fonction correspondant dans votre compte.
3. Lorsque vous exécutez une requête DML ou DDL Athena qui utilise le nom du catalogue, le moteur de requête Athena appelle le nom de fonction Lambda que vous avez associé au nom du catalogue.
4. À l'aide de AWS PrivateLink, la fonction Lambda communique avec le métastore Hive externe de votre VPC et reçoit des réponses aux demandes de métadonnées. Athena utilise les métadonnées de votre métastore Hive externe tout comme il utilise les métadonnées du AWS Glue Data Catalog par défaut.

Considérations et restrictions

Lorsque vous utilisez le connecteur de données Athena pour le métastore Hive externe, tenez compte des points suivants :

- Vous pouvez utiliser l'instruction CTAS pour créer une table sur un métastore Hive externe.
- Vous pouvez utiliser l'instruction INSERT INTO pour insérer des données dans un métastore Hive externe.
- La prise en charge DDL pour le métastore Hive externe est limitée aux instructions suivantes.
  - ALTER DATABASE SET DBPROPERTIES
  - ALTER TABLE ADD COLUMNS
  - ALTER TABLE ADD PARTITION
  - ALTER TABLE DROP PARTITION
  - ALTER TABLE RENAME PARTITION
  - ALTER TABLE REPLACE COLUMNS
  - ALTER TABLE SET LOCATION
  - ALTER TABLE SET TBLPROPERTIES
  - CREATE DATABASE
  - CREATE TABLE
  - CREATE TABLE AS
  - DESCRIBE TABLE
  - DROP DATABASE
  - DROP TABLE
  - SHOW COLUMNS
  - SHOW CREATE TABLE
  - SHOW PARTITIONS
  - SHOW SCHEMAS
  - SHOW TABLES
  - SHOW TBLPROPERTIES
- Le nombre maximal de catalogues enregistrés que vous pouvez avoir est de 1 000.
- L'authentification Kerberos pour le métastore Hive n'est pas prise en charge.

- Pour utiliser le pilote JDBC avec un metastore Hive externe ou des requêtes fédérées, incluez `MetadataRetrievalMethod=ProxyAPI` dans votre chaîne de connexion JDBC. Pour plus d'informations sur le pilote JDBC, voir Connexion à Amazon Athena avec JDBC.
- Les colonnes cachées `$path`, `$bucket`, `$file_size`, `$file_modified_time`, `$partition`, `$row_id` de Hive ne peuvent pas être utilisées pour le filtrage du contrôle d'accès précis.
- Les tables cachées du système Hive comme *`example_table`*`$partitions` ou *`example_table`*`$properties` ne sont pas prises en charge par le contrôle d'accès précis.

Autorisations

Les connecteurs de données prédéfinis et personnalisés peuvent nécessiter l'accès aux ressources suivantes pour fonctionner correctement. Vérifiez les informations relatives au connecteur que vous utilisez pour vous assurer que vous avez correctement configuré votre VPC. Pour de plus amples informations sur les autorisations IAM requises pour exécuter des requêtes et créer un connecteur de source de données dans Athena, voir Autorisation d'accès à un connecteur de données Athena pour un métastore Hive externe et Autorisation d'accès des fonctions Lambda aux métastores Hive externes.

- Simple Storage Service (Amazon S3) – Outre l'écriture des résultats des requêtes dans l'emplacement des résultats des requêtes Athena dans Simple Storage Service (Amazon S3), les connecteurs de données écrivent également dans un compartiment de déversement dans Simple Storage Service (Amazon S3). Une connectivité et des autorisations d'accès à cet emplacement Simple Storage Service (Amazon S3) sont requises. Pour plus d'informations, consultez Emplacement de déversement dans Simple Storage Service (Amazon S3) plus loin dans cette rubrique.
- Athena – L'accès est nécessaire pour vérifier l'état de la requête et empêcher le surbalayage.
- AWS Glue— L'accès est requis si votre connecteur utilise AWS Glue des métadonnées supplémentaires ou principales.
- AWS Key Management Service
- Politiques – Le métastore Hive, Athena Query Federation et les UDF nécessitent des politiques en plus de AWS politique gérée : AmazonAthenaFullAccess. Pour plus d'informations, consultez Gestion des identités et des accès dans Athena.

Emplacement de déversement dans Simple Storage Service (Amazon S3)

En raison de la limite de la taille des réponses des fonctions Lambda, les réponses supérieures au seuil sont déversées dans un emplacement Simple Storage Service (Amazon S3) que vous spécifiez lorsque vous créez votre fonction Lambda. Athena lit directement ces réponses de Simple Storage Service (Amazon S3).

> **Note**
>
> Athena ne supprime pas les fichiers de réponse sur Simple Storage Service (Amazon S3). Nous vous recommandons de configurer une politique de rétention pour supprimer automatiquement les fichiers de réponse.

Connexion d'Athena à un métastore Apache Hive

Pour connecter Athena à un métastore Apache Hive, vous devez créer et configurer une fonction Lambda. Pour une implémentation de base, vous pouvez effectuer toutes les étapes requises à partir de la console de gestion Athena.

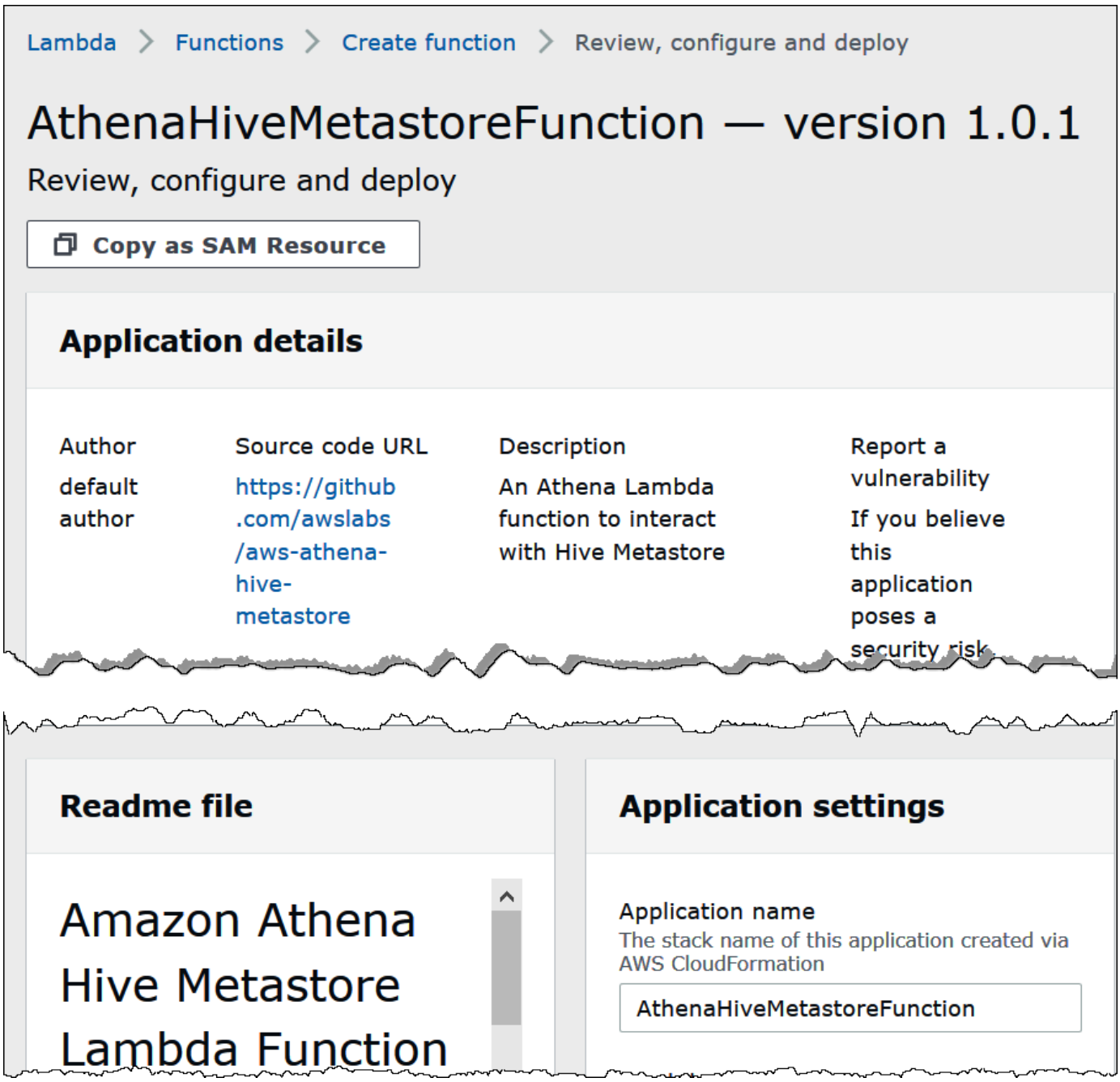
> **Note**
>
> La procédure suivante nécessite que vous ayez l'autorisation de créer un rôle IAM personnalisé pour la fonction Lambda. Si vous n'êtes pas autorisé à créer un rôle personnalisé, vous pouvez utiliser l'implémentation de référence Athena pour créer une fonction Lambda séparément, puis utiliser la AWS Lambda console pour choisir un rôle IAM existant pour la fonction. Pour plus d’informations, consultez Connexion d'Athena à un métastore Hive en utilisant un rôle d'exécution IAM existant.

Connexion d'Athena à un métastore Hive

1. Ouvrez la console Athena à l’adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Choisissez Sources de données.
4. Dans le coin supérieur droit de la console, choisissez Create data source (Créer une source de données).
5. Sur la page Choose a data sources (Choisir une sources de données), pour Data source (Source de données), choisissez S3 - Apache Hive metastore (S3 - Métastore Apache Hive).
6. Choisissez Suivant.
7. Dans la section Data source details (Détails sur la source de données), pour Data source name (Nom de la source de données), saisissez le nom que vous souhaitez utiliser dans vos instructions SQL lorsque vous interrogez la source de données à partir d'Athena. Le nom peut contenir jusqu'à 127 caractères et doit être unique dans votre compte. Il ne peut pas être modifié après sa création. Les caractères valides sont a-z, A-Z, 0–9, _ (trait de soulignement), @ (arobase) et - (trait d'union). Les noms `awsdatacatalog`, `hive`, `jmx` et `system` sont réservés par Athena et ne peuvent pas être utilisés pour les noms de source de données.
8. Pour la fonction Lambda, choisissez Create Lambda function, puis Create a new Lambda function dans AWS Lambda

   La AthenaHiveMetastoreFunctionpage s'ouvre dans la AWS Lambda console. La page contient des informations détaillées sur le connecteur.

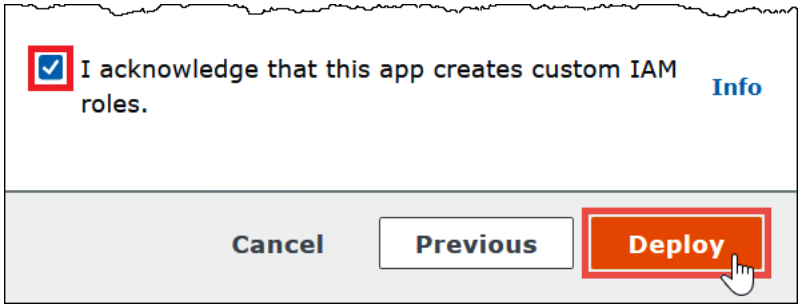


9. Sous Application settings (Paramètres de l'application), saisissez les paramètres de votre fonction Lambda.

   - LambdaFuncName— Donnez un nom à la fonction. Par exemple, myHiveMetastore.
   - SpillLocation— Spécifiez un emplacement Amazon S3 dans ce compte pour stocker les métadonnées dérivées si la taille de réponse de la fonction Lambda dépasse 4 Mo.
   - HMSUris – Saisissez l'URI de l'hôte de votre métastore Hive qui utilise le protocole Thrift au port 9083. Utilisez la syntaxe `thrift://<host_name>:9083`.

- LambdaMemory— Spécifiez une valeur comprise entre 128 Mo et 3 008 Mo. La fonction Lambda se voit allouer des cycles d'UC proportionnels à la quantité de mémoire que vous configurez. La valeur par défaut est 1024.
- LambdaTimeout— Spécifiez le temps d'exécution d'appel Lambda maximal autorisé en secondes, de 1 à 900 (900 secondes correspondent à 15 minutes). La valeur par défaut est de 300 secondes (5 minutes).
- VPC SecurityGroupIds — Entrez une liste d'identifiants de groupes de sécurité VPC séparés par des virgules pour le métastore Hive.
- VPC SubnetIds — Entrez une liste d'ID de sous-réseau VPC séparés par des virgules pour le métastore Hive.

10. Sélectionnez I acknowledge that this app creates custom IAM roles (Je comprends que cette application crée des rôles IAM personnalisés) puis choisissez Deploy (Déployer).



Une fois le déploiement terminé, votre fonction apparaît dans votre liste d'applications Lambda. Maintenant que la fonction de métastore Hive a été déployée sur votre compte, vous pouvez configurer Athena pour l'utiliser.

11. Revenez à la page Enter data sources details (Saisir les détails des sources de données) de la console Athena.
12. Dans la section Lambda function (fonction Lambda), choisissez l'icône d'actualisation située à côté de la zone de recherche de fonction Lambda. L'actualisation de la liste des fonctions disponibles entraîne l'apparition de la fonction nouvellement créée dans la liste.
13. Choisissez le nom de la fonction que vous venez de créer dans la console Lambda. L'ARN de la fonction Lambda s'affiche.
14. (Facultatif) Pour Tags (Identifications), ajoutez des paires clé-valeur à associer à cette source de données. Pour en savoir plus sur les identifications, consultez Étiquetage des ressources Athena.
15. Choisissez Suivant.

16. À la page Review and create, vérifiez les détails de la source de données, puis choisissez Create data source (Créer une source de données).
17. La section Data source details (Détails de source de données) de la page de votre source de données affiche des informations sur votre nouveau connecteur.

    Vous pouvez désormais utiliser le nom de la source de données que vous avez spécifié pour référencer le métastore Hive dans vos requêtes SQL dans Athena. Dans vos requêtes SQL, utilisez l'exemple de syntaxe suivant, en remplaçant `hms-catalog-1` par le nom de catalogue que vous avez spécifié précédemment.

    ```
    SELECT * FROM hms-catalog-1.CustomerData.customers
    ```
18. Pour plus d'informations sur l'affichage, la modification ou la suppression des sources de données que vous créez, consultez Gestion des sources de données.

Utilisation du connecteur AWS Serverless Application Repository de source de données Hive pour déployer un

Pour déployer un connecteur de source de données Athena pour Hive, vous pouvez utiliser le AWS Serverless Application Repository au lieu de commencer par la console Athena. Utilisez le AWS Serverless Application Repository pour trouver le connecteur que vous souhaitez utiliser, fournir les paramètres requis par le connecteur, puis déployez le connecteur sur votre compte. Ensuite, après avoir déployé le connecteur, vous utilisez la console Athena pour rendre la source de données disponible pour Athena.

Pour utiliser le AWS Serverless Application Repository pour déployer un connecteur de source de données pour Hive sur votre compte

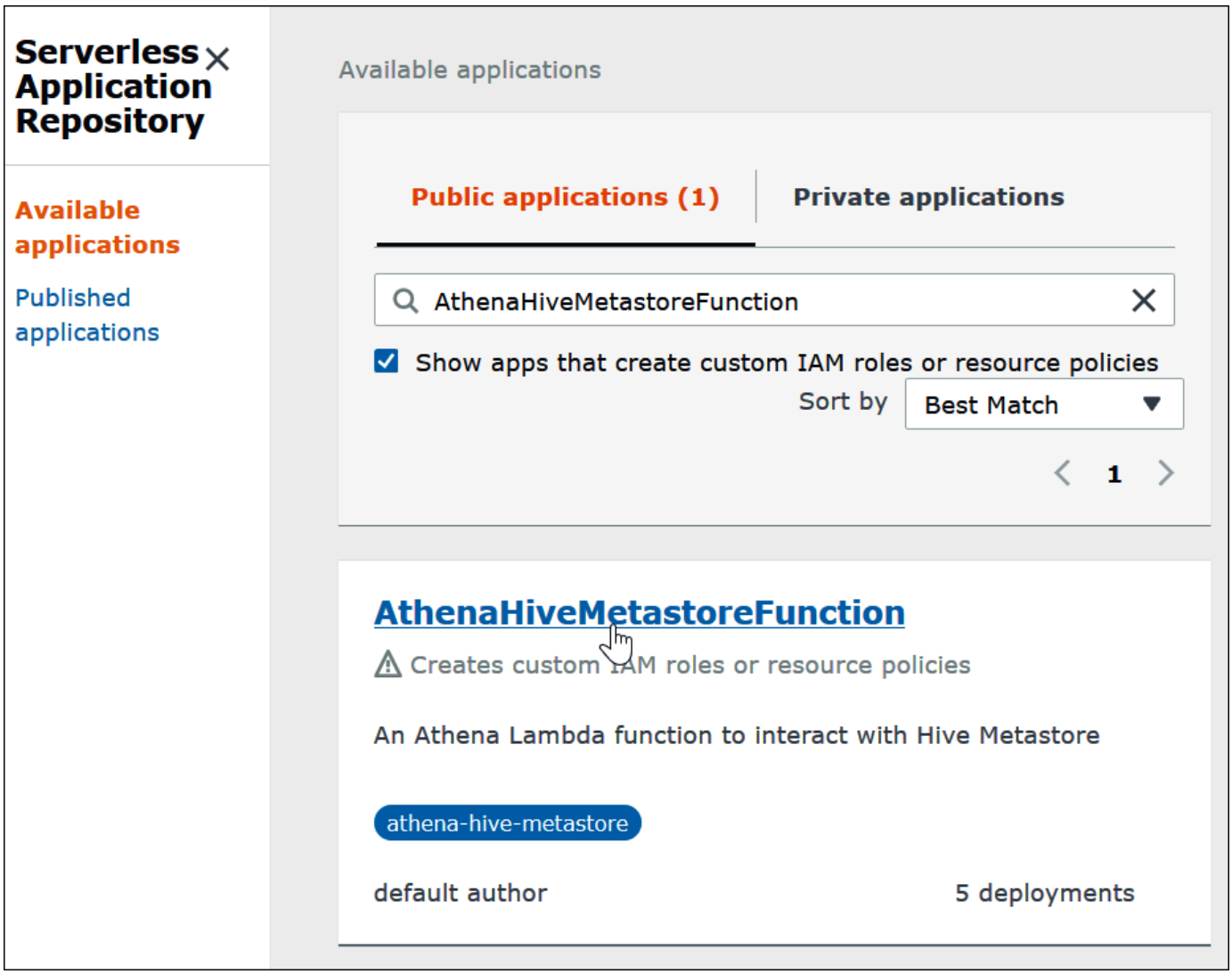
1. Connectez-vous au référentiel d'applications sans serveur AWS Management Console et ouvrez-le.
2. Dans le volet de navigation, choisissez Applications.
3. Sélectionnez l'option Show apps that create custom IAM roles or resource policies (Afficher les applications qui créent des rôles IAM ou des politiques de ressources personnalisés).
4. Dans la zone de recherche, saisissez **`Hive`**. Les connecteurs qui apparaissent incluent les deux connecteurs suivants :

    - AthenaHiveMetastoreFunction – Fichier `.jar` de fonction Lambda uber.

- AthenaHiveMetastoreFunctionWithLayer— Couche Lambda et fichier de fonctions Lambda fin. `.jar`

Les deux applications ont les mêmes fonctionnalités et ne diffèrent que par leur implémentation. Vous pouvez utiliser l'une de ces deux méthodes pour créer une fonction Lambda qui connecte Athena à votre métastore Hive.

5. Choisissez le nom du connecteur que vous souhaitez utiliser. Ce tutoriel utilise AthenaHiveMetastoreFunction.



6. Sous Application settings (Paramètres de l'application), saisissez les paramètres de votre fonction Lambda.

- LambdaFuncName— Donnez un nom à la fonction. Par exemple, myHiveMetastore.

- SpillLocation— Spécifiez un emplacement Amazon S3 dans ce compte pour stocker les métadonnées dérivées si la taille de réponse de la fonction Lambda dépasse 4 Mo.
- HMSUris – Saisissez l'URI de l'hôte de votre métastore Hive qui utilise le protocole Thrift au port 9083. Utilisez la syntaxe `thrift://<host_name>:9083`.
- LambdaMemory— Spécifiez une valeur comprise entre 128 Mo et 3 008 Mo. La fonction Lambda se voit allouer des cycles d'UC proportionnels à la quantité de mémoire que vous configurez. La valeur par défaut est 1024.
- LambdaTimeout— Spécifiez le temps d'exécution d'appel Lambda maximal autorisé en secondes, de 1 à 900 (900 secondes correspondent à 15 minutes). La valeur par défaut est de 300 secondes (5 minutes).
- VPC SecurityGroupIds — Entrez une liste d'identifiants de groupes de sécurité VPC séparés par des virgules pour le métastore Hive.
- VPC SubnetIds — Entrez une liste d'ID de sous-réseau VPC séparés par des virgules pour le métastore Hive.

7. En bas à droite de la page Application details (Détails de l'application), sélectionnez I acknowledge that this app creates custom IAM roles (Je reconnais que cette application crée des rôles IAM personnalisés), puis choisissez Deploy (Déployer).

À ce stade, vous pouvez configurer le service Athena pour qu'il utilise votre fonction Lambda afin de se connecter à votre métastore Hive. Pour les étapes, consultez Configuration d'Athena pour utiliser un connecteur de métastore Hive déployé.

Connexion d'Athena à un métastore Hive en utilisant un rôle d'exécution IAM existant

Pour connecter votre métastore Hive externe à Athena avec une fonction Lambda qui utilise un rôle IAM existant, vous pouvez utiliser l'implémentation de référence du connecteur Athena pour métastore Hive externe.

Les trois étapes principales sont les suivantes :

1. Cloner et créer – Clonez l'implémentation de référence Athena et créez le fichier JAR qui contient le code de la fonction Lambda.
2. AWS Lambda console — Dans la AWS Lambda console, créez une fonction Lambda, attribuez-lui un rôle d'exécution IAM existant et téléchargez le code de fonction que vous avez généré.

3. Console Amazon Athena – Dans la console Amazon Athena, créez un nom de source de données que vous pourrez utiliser pour faire référence à votre métastore Hive externe dans vos requêtes Athena.

Si vous êtes déjà autorisé à créer un rôle IAM personnalisé, vous pouvez utiliser un flux de travail plus simple qui utilise la console Athena pour créer et AWS Serverless Application Repository configurer une fonction Lambda. Pour plus d'informations, consultez Connexion d'Athena à un métastore Apache Hive.

Prérequis

- Git doit être installé sur votre système.
- Vous devez avoir Apache Maven installé.
- Vous avez un rôle d'exécution IAM que vous pouvez attribuer à la fonction Lambda. Pour plus d'informations, consultez Autorisation d'accès des fonctions Lambda aux métastores Hive externes.

Clonage et création de la fonction Lambda

Le code de fonction pour l'implémentation de référence Athena est un projet Maven situé sur GitHub awslabs/. aws-athena-hive-metastore Pour des informations détaillées sur le projet, consultez le fichier README correspondant GitHub ou le Implémentation de référence sujet de cette documentation.

Clonage et création du code de fonction Lambda

1. Saisissez la commande suivante pour cloner l'implémentation de référence Athena :

```
git clone https://github.com/awslabs/aws-athena-hive-metastore
```

2. Exécutez la commande suivante pour créer le fichier `.jar` pour la fonction Lambda :

```
mvn clean install
```

Une fois le projet crée avec succès, le fichier `.jar` suivant est créé dans le dossier cible de votre projet :

`hms-lambda-func-1.0-SNAPSHOT-withdep.jar`

Dans la section suivante, vous allez utiliser la AWS Lambda console pour télécharger ce fichier sur votre compte Amazon Web Services.

Création et configuration de la fonction Lambda dans la console AWS Lambda

Dans cette section, vous allez utiliser la AWS Lambda console pour créer une fonction qui utilise un rôle d'exécution IAM existant. Après avoir configuré un VPC pour la fonction, vous téléchargez le code de la fonction et configurez les variables d'environnement pour la fonction.

Créer la fonction Lambda

Au cours de cette étape, vous créez une fonction dans la AWS Lambda console qui utilise un rôle IAM existant.

Création d'une fonction Lambda qui utilise un rôle IAM existant

1. Connectez-vous à la AWS Lambda console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lambda/.
2. Dans le volet de navigation, choisissez Fonctions.
3. Choisissez Créer une fonction.
4. Choisissez Créer à partir de zéro.
5. Dans Function name (Nom de la fonction), saisissez le nom de votre fonction Lambda (par exemple, **`EHMSBasedLambda`**).
6. Pour Runtime (Exécution), choisissez Java 8.
7. Sous Permissions (Autorisations), développez Change default execution role (Modifier le rôle d'exécution par défaut).
8. Pour Execution role (Rôle d'exécution), choisissez Use an existing role (Utilisez un rôle existant).
9. Pour Existing role (Rôle existant), choisissez le rôle d'exécution IAM que votre fonction Lambda utilisera pour Athena (cet exemple utilise un rôle appelé `AthenaLambdaExecutionRole`).
10. Développez Advanced settings (Paramètres avancés).
11. Sélectionnez Enable Network (Activer le réseau).
12. Pour VPC, choisissez le VPC auquel votre fonction aura accès.
13. Pour Subnets (Sous-réseaux), choisissez les sous-réseaux VPC que Lambda doit utiliser.
14. Pour Security groups (Groupe de sécurité), choisissez les groupes de sécurité VPC à utiliser pour Lambda.

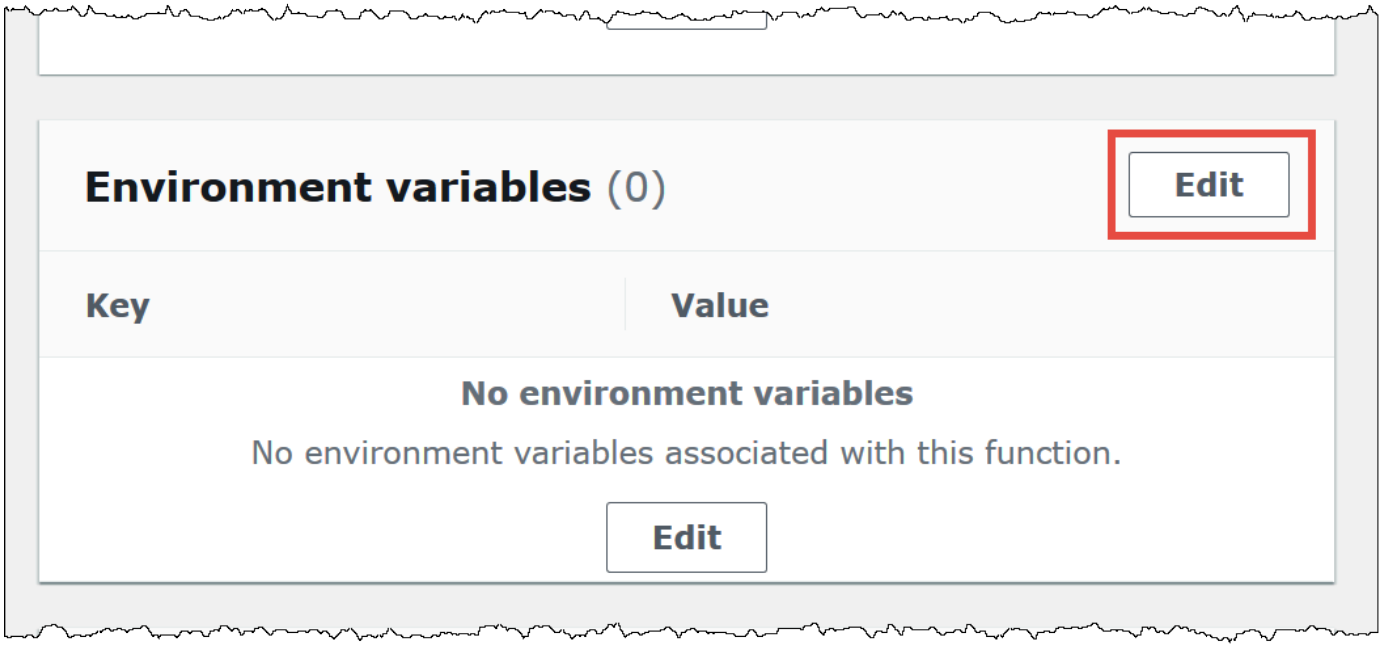
15. Sélectionnez Create function (Créer une fonction). La AWS Lambda console ouvre la page de configuration de votre fonction et commence à créer votre fonction.

Chargement du code et configuration de la fonction Lambda

Lorsque la console vous informe que votre fonction a été créée avec succès, vous pouvez télécharger le code de la fonction et configurer ses variables d'environnement.

Téléchargement du code de votre fonction Lambda et configuration de ses variables d'environnement

1. Dans la console Lambda, assurez-vous que vous êtes sur l'onglet Code de la page de la fonction que vous avez spécifiée.
2. Pour Code source (Code source), choisissez Upload from (Charger depuis) puis choisissez .zip or .jar file (fichier .zip ou .jar).
3. Téléchargement du fichier `hms-lambda-func-1.0-SNAPSHOT-withdep.jar` généré précédemment.
4. Sur la page de la fonction Lambda, choisissez l'onglet Configuration.
5. Dans le panneau de gauche, choisissez Environment variables (Variables d'environnement).
6. Dans la section Environment variables (Variables d'environnement), choisissez Edit (Modifier).

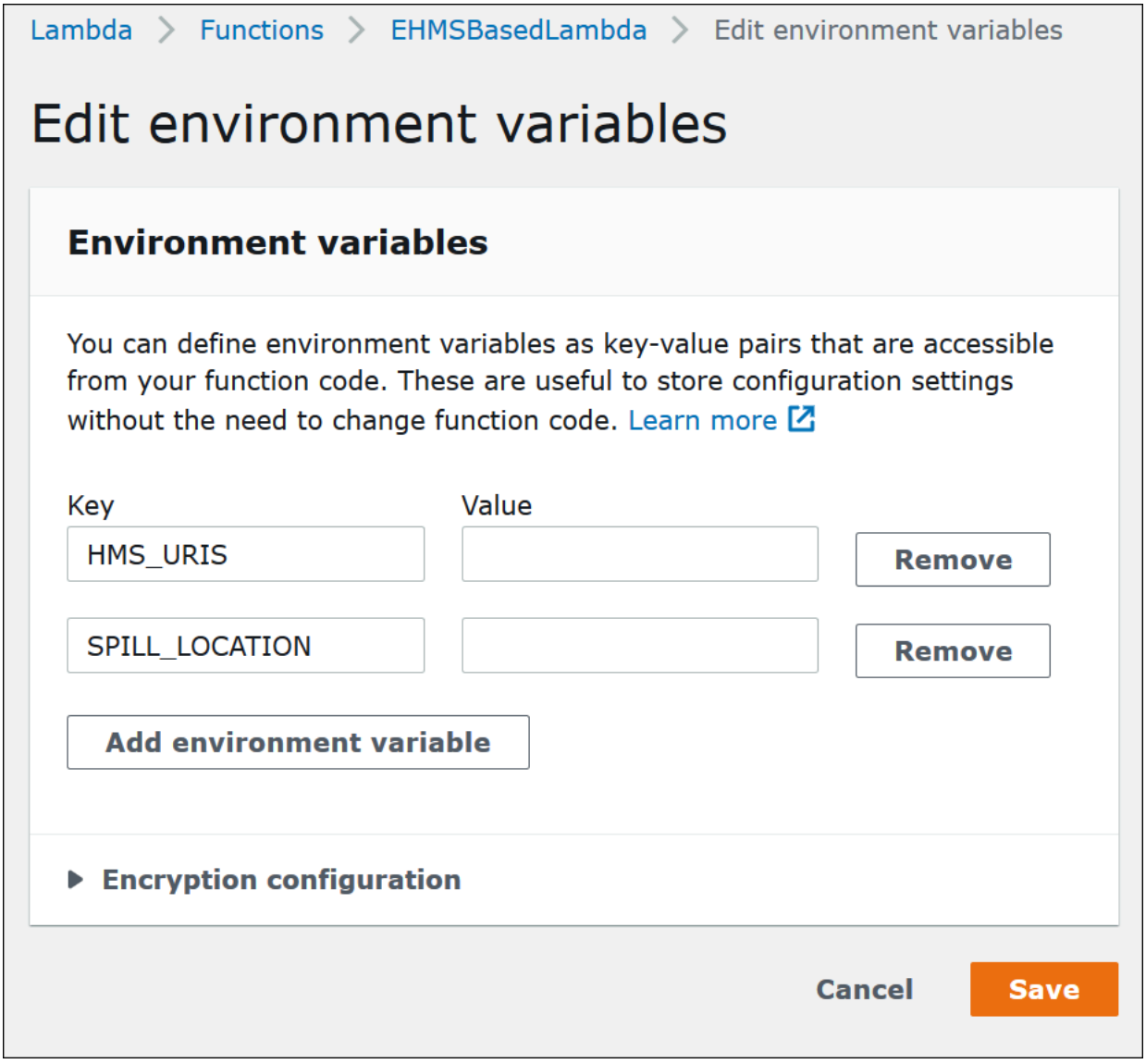


7. Sur la page Edit environment variables (Modifier les variables d'environnement), utilisez l'option Add environment variable (Ajouter une variable d'environnement) pour ajouter les clés et les valeurs des variables d'environnement suivantes :

- HMS_URIS – Utilisez la syntaxe suivante pour saisir l'URI de votre hôte de métastore Hive qui utilise le protocole Thrift au port 9083.

```
thrift://<host_name>:9083
```

- SPILL_LOCATION – Spécifiez un emplacement Amazon S3 dans votre compte Amazon Web Services pour contenir les métadonnées de débordement si la taille de la réponse de la fonction Lambda dépasse 4 Mo.



8. Choisissez Enregistrer.

À ce stade, vous êtes prêt à configurer Athena pour qu'il utilise votre fonction Lambda afin de se connecter à votre métastore Hive. Pour les étapes, consultez Configuration d'Athena pour utiliser un connecteur de métastore Hive déployé.

Configuration d'Athena pour utiliser un connecteur de métastore Hive déployé

Une fois que vous avez déployé un connecteur de source de données Lambda comme `AthenaHiveMetastoreFunction` sur votre compte, vous pouvez configurer Athena pour l'utiliser. Pour ce faire, créez un nom de source de données qui fait référence à votre métastore Hive externe à utiliser dans vos requêtes Athena.

Connexion d'Athena à votre métastore Hive à l'aide d'une fonction Lambda existante

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Choisissez Sources de données.
4. À la page Data sources (Sources de données), choisissez Connect data source (Connecter la source de données).
5. Sur la page Choose a data sources (Choisir une sources de données), pour Data source (Source de données), choisissez S3 - Apache Hive metastore (S3 - Métastore Apache Hive).
6. Choisissez Suivant.
7. Dans la section Data source details (Détails sur la source de données), pour Data source name (Nom de la source de données), saisissez le nom que vous souhaitez utiliser dans vos

instructions SQL lorsque vous interrogez la source de données à partir d'Athena (par exemple, `MyHiveMetastore`). Le nom peut contenir jusqu'à 127 caractères et doit être unique dans votre compte. Il ne peut pas être modifié après sa création. Les caractères valides sont a-z, A-Z, 0–9, _ (trait de soulignement), @ (arobase) et - (trait d'union). Les noms `awsdatacatalog`, `hive`, `jmx` et `system` sont réservés par Athena et ne peuvent pas être utilisés pour les noms de source de données.

8. Dans la section Connection details (Détails de connexion), utilisez la zone Select or enter a Lambda function (Sélectionner ou saisir une fonction Lambda) pour choisir le nom de la fonction que vous venez de créer. L'ARN de la fonction Lambda s'affiche.
9. (Facultatif) Pour Tags (Identifications), ajoutez des paires clé-valeur à associer à cette source de données. Pour en savoir plus sur les identifications, consultez Étiquetage des ressources Athena.
10. Choisissez Suivant.
11. À la page Review and create, vérifiez les détails de la source de données, puis choisissez Create data source (Créer une source de données).
12. La section Data source details (Détails de source de données) de la page de votre source de données affiche des informations sur votre nouveau connecteur.

    Vous pouvez désormais utiliser le nom de la source de données que vous avez spécifié pour référencer le métastore Hive dans vos requêtes SQL dans Athena.

    Dans vos requêtes SQL, utilisez l'exemple de syntaxe suivant, en remplaçant `ehms-catalog` par le nom de la source de données que vous avez spécifié précédemment.

    ```
    SELECT * FROM ehms-catalog.CustomerData.customers
    ```
13. Pour afficher, modifier ou supprimer les sources de données que vous créez, veuillez consulter Gestion des sources de données.

Utilisation d'un nom de source de données par défaut dans des requêtes de métastore Hive externes

Lorsque vous exécutez des requêtes DML et DDL sur des métastores Hive externes, vous pouvez simplifier la syntaxe de la requête en omettant le nom du catalogue si ce nom est sélectionné dans l'éditeur de requête. Certaines restrictions s'appliquent à cette fonctionnalité.

Instructions DML

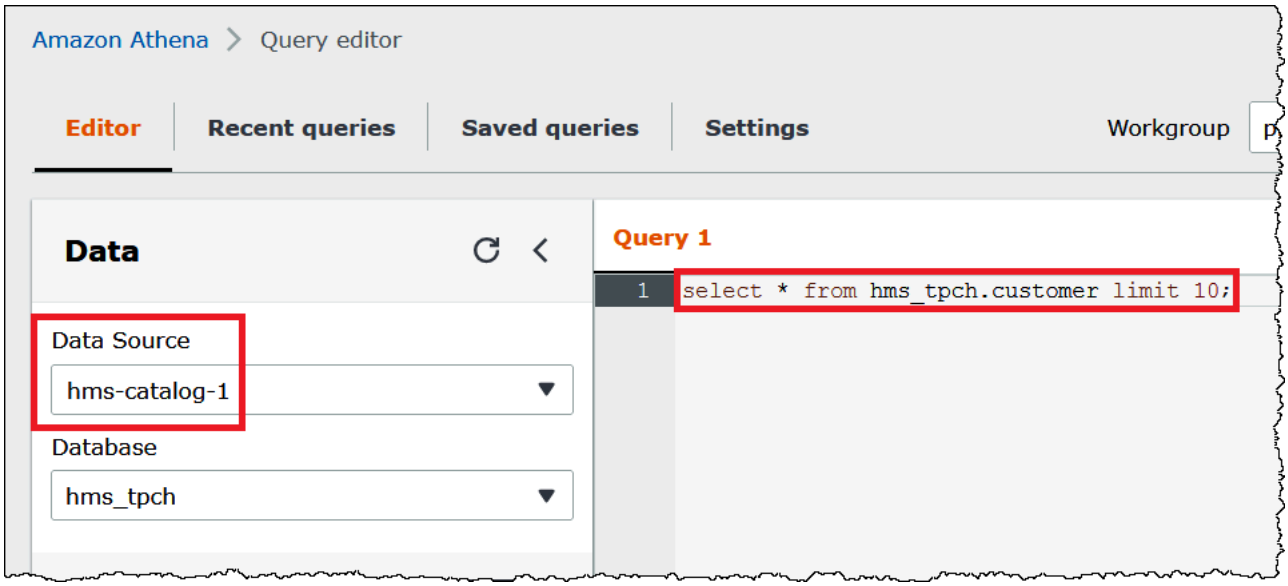
Pour exécuter des requêtes avec des catalogues enregistrés

1. Vous pouvez placer le nom de la source de données avant la base de données en utilisant la syntaxe `[[data_source_name].database_name].table_name`, comme dans l'exemple suivant.

```
select * from  "hms-catalog-1".hms_tpch.customer limit 10;
```

2. Lorsque la source de données que vous souhaitez utiliser est déjà sélectionnée dans l'éditeur de requêtes, vous pouvez omettre le nom dans la requête, comme dans l'exemple suivant.

```
select * from hms_tpch.customer limit 10:
```



3. Lorsque vous utilisez plusieurs sources de données dans une requête, vous pouvez omettre uniquement le nom de la source de données par défaut et devez spécifier le nom complet de toutes les autres sources de données qui ne sont pas des sources de données par défaut.

   Supposons par exemple que `AwsDataCatalog` est sélectionnée comme source de données par défaut dans l'éditeur de requêtes. L'`FROM`instruction contenue dans l'extrait de requête suivant qualifie entièrement les deux premiers noms de source de données, mais omet le nom de la troisième source de données car elle figure dans le catalogue de AWS Glue données.

```
...
FROM ehms01.hms_tpch.customer,
```

```
        "hms-catalog-1".hms_tpch.orders,
        hms_tpch.lineitem
...
```

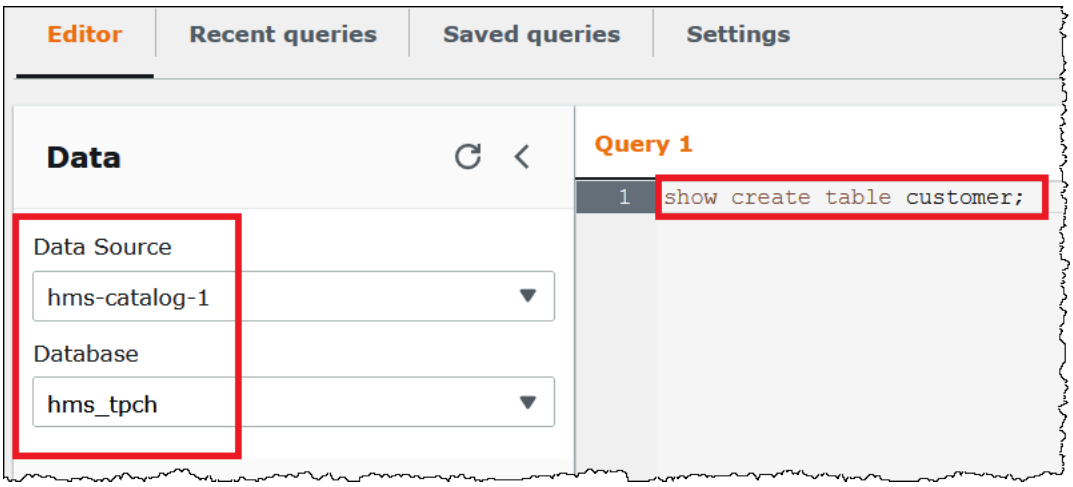
Instructions DDL

Les instructions DDL Athena suivantes prennent en charge les préfixes de nom de catalogue. Les préfixes de nom de catalogue dans d'autres instructions DDL provoquent des erreurs de syntaxe.

```
SHOW TABLES [IN [catalog_name.]database_name] ['regular_expression']

SHOW TBLPROPERTIES [[catalog_name.]database_name.]table_name [('property_name')]

SHOW COLUMNS IN [[catalog_name.]database_name.]table_name

SHOW PARTITIONS [[catalog_name.]database_name.]table_name

SHOW CREATE TABLE [[catalog_name.][database_name.]table_name

DESCRIBE [EXTENDED | FORMATTED] [[catalog_name.][database_name.]table_name [PARTITION
 partition_spec] [col_name ( [.field_name] | [.'$elem$'] | [.'$key$'] | [.'$value$'] )]
```

Comme pour les instructions DML, vous pouvez omettre la source de données et les préfixes de base de données de la requête lorsque la source de données et la base de données sont sélectionnées dans l'éditeur de requêtes.

Dans l'image suivante, la source de données `hms-catalog-1` et la base de données `hms_tpch` sont sélectionnées dans l'éditeur de requêtes. L'instruction `show create table customer` réussit, même si le préfixe `hms-catalog-1` et le nom de base de données `hms_tpch` sont omis de la requête elle-même.



Spécification d'une source de données par défaut dans une chaîne de connexion JDBC

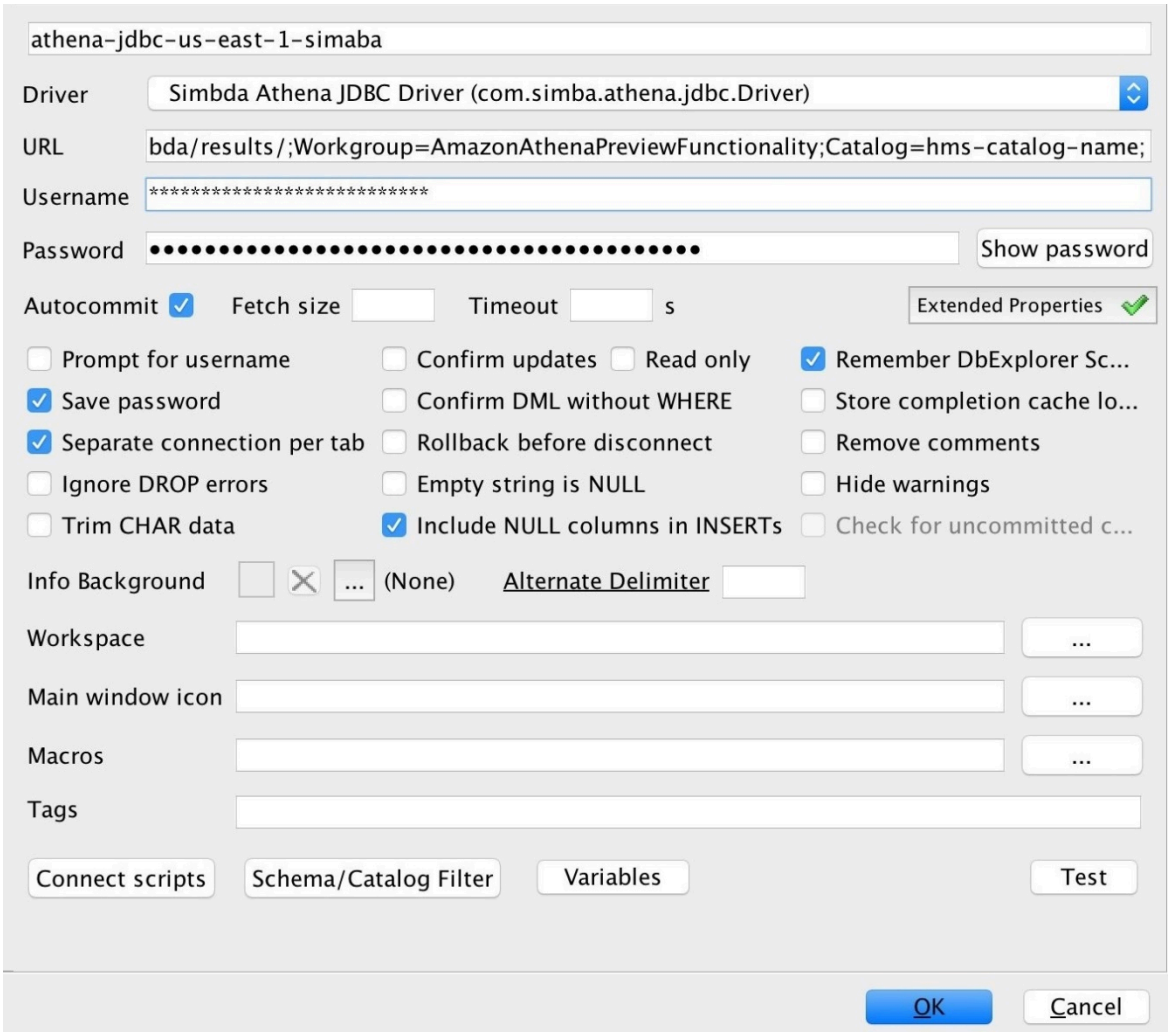
Lorsque vous utilisez le pilote JDBC Athena pour connecter Athena à un métastore Hive externe, vous pouvez utiliser le paramètre `Catalog` pour spécifier le nom de la source de données par défaut dans votre chaîne de connexion dans un éditeur SQL tel que SQL Workbench.

> **Note**
>
> Pour télécharger les pilotes les plus récents d'Athena JDBC, consultez Utilisation d'Athena avec le pilote JDBC.

La chaîne de connexion suivante indique la source de données par défaut *hms-catalog-name*.

```
    jdbc:awsathena://AwsRegion=us-east-1;S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/
lambda/results/;Workgroup=AmazonAthenaPreviewFunctionality;Catalog=hms-catalog-name;
```

L'image suivante illustre un URL de connexion JDBC configuré dans SQL Workbench.



Utilisation des vues Hive

Vous pouvez utiliser Athena pour interroger les vues existantes dans vos métastores Hive externes. Athena traduit vos vues pour vous on-the-fly lors de l'exécution sans modifier la vue d'origine ni enregistrer la traduction.

Par exemple, supposons que vous disposez d'une vue Hive comme la suivante qui utilise une syntaxe non prise en charge par Athena comme `LATERAL VIEW explode()` :

```
CREATE VIEW team_view AS
SELECT team, score
FROM matches
LATERAL VIEW explode(scores) m AS score
```

Athena traduit la chaîne de requête de la vue Hive en une instruction semblable à la suivante qu'Athena peut exécuter :

```
SELECT team, score
FROM matches
CROSS JOIN UNNEST(scores) AS m (score)
```

Pour plus d'informations sur la connexion d'un métastore Hive externe à Athena, consultez Utilisation du connecteur de données Athena pour un métastore Hive externe.

Considérations et restrictions

Lorsque vous interrogez les vues Hive d'Athena, tenez compte des points suivants :

- Athena ne prend pas en charge la création de vues Hive. Vous pouvez créer des vues Hive dans votre métastore Hive externe, que vous pouvez ensuite interroger depuis Athena.
- Athena ne prend pas en charge les fonctions UDF personnalisées pour les vues Hive.
- En raison d'un problème connu dans la console Athena, les vues Hive apparaissent sous la liste des tables au lieu de la liste des vues.
- Bien que le processus de traduction soit automatique, certaines fonctions Hive ne sont pas prises en charge pour les vues Hive ou nécessitent un traitement spécial. Pour plus d'informations, consultez la section suivante.

Limitations du support de la fonction Hive

Cette section met en évidence les fonctions Hive qu'Athena ne prend pas en charge pour les vues Hive ou qui nécessitent un traitement spécial. Actuellement, étant donné qu'Athena prend principalement en charge les fonctions de Hive 2.2.0, les fonctions disponibles uniquement dans les versions supérieures (telles que Hive 4.0.0) ne sont pas disponibles. Pour obtenir la liste complète des fonctions Hive, consultez les fonctions définies par l'utilisateur LanguageManual Hive.

Fonctions d’agrégation

Fonctions d'agrégation qui nécessitent un traitement spécial

La fonction d'agrégation suivante pour les vues Hive nécessite un traitement spécial.

- Avg– Au lieu de `avg(INT i)`, utilisez `avg(CAST(i AS DOUBLE))`.

Fonctions d'agrégation non prises en charge

Les fonctions d'agrégation Hive suivantes ne sont pas prises en charge dans les vues Athena pour Hive.

```
covar_pop
histogram_numeric
ntile
percentile
percentile_approx
```

Les fonctions de régression telles que `regr_count`, `regr_r2`, et `regr_sxx` ne sont pas prises en charge dans les vues Athena pour Hive.

Fonctions de date non prises en charge

Les fonctions de date Hive suivantes ne sont pas prises en charge dans les vues Athena pour Hive.

```
date_format(date/timestamp/string ts, string fmt)
day(string date)
dayofmonth(date)
extract(field FROM source)
hour(string date)
minute(string date)
month(string date)
quarter(date/timestamp/string)
second(string date)
weekofyear(string date)
year(string date)
```

Fonctions de masquage non prises en charge

Les fonctions de masquage Hive telles que `mask()` et `mask_first_n()` ne sont pas pris en charge dans les vues Athena pour Hive.

Fonctions diverses

Fonctions diverses qui nécessitent un traitement spécial

Les fonctions diverses suivantes pour les vues Hive nécessitent un traitement spécial.

- md5 – Athena prend en charge `md5(binary)`, mais pas `md5(varchar)`.
- Explose – Athena prend en charge `explode` lorsqu'il est utilisé dans la syntaxe suivante :

```
LATERAL VIEW [OUTER] EXPLODE(<argument>)
```

- Posexplose – Athena prend en charge `posexplode` lorsqu'il est utilisé dans la syntaxe suivante :

```
LATERAL VIEW [OUTER] POSEXPLODE(<argument>)
```

  Dans la sortie `(pos, val)`, Athena traite la colonne `pos` comme `BIGINT`. Pour cette raison, vous devrez peut-être convertir la colonne `pos` en `BIGINT` pour éviter une vue obsolète. L'exemple suivant illustre cette technique.

```
SELECT CAST(c AS BIGINT) AS c_bigint, d
FROM table LATERAL VIEW POSEXPLODE(<argument>) t AS c, d
```

Fonctions diverses non prises en charge

Les fonctions Hive suivantes ne sont pas prises en charge dans les vues Athena pour Hive.

```
aes_decrypt
aes_encrypt
current_database
current_user
inline
java_method
logged_in_user
reflect
sha/sha1/sha2
stack
version
```

Opérateurs

Opérateurs nécessitant un traitement spécial

Les opérateurs suivants pour les vues Hive nécessitent un traitement spécial.

- Opérateur Mod (%) – Parce que le type `DOUBLE` convertit implicitement en `DECIMAL(x,y)`, la syntaxe suivante peut entraîner un message d'erreur View is stale (La vue est obsolète) :

```
a_double % 1.0 AS column
```

Pour résoudre ce problème, utilisez CAST, comme dans l'exemple suivant.

```
CAST(a_double % 1.0 as DOUBLE) AS column
```

- Opérateur de division (/) – Dans Hive, `int` divisé par `int` produit un `double`. Dans Athena, la même opération produit un `int` tronqué.

Opérateurs non pris en charge

Athena ne prend pas en charge les opérateurs suivants pour les vues Hive.

~A – au niveau du bit NOT

A ^ b – au niveau du bit XOR

A & b – au niveau du bit AND

A | b – au niveau du bit OR

A **<=>** b – renvoie le même résultat que l'opérateur égal (=) pour opérandes non nuls. Renvoie TRUE si les deux sont NULL, FALSE si l'un d'eux est NULL.

Fonctions de chaîne

Fonctions de chaîne qui nécessitent un traitement spécial

Les fonctions de chaîne Hive suivantes pour les vues Hive nécessitent un traitement spécial.

- chr(bigint|double a) – Hive autorise des arguments négatifs ; pas Athena.
- instr(string str, string substr) – Parce que le mappage d'Athena pour la fonction `instr` renvoie BIGINT et non INT, utilisez la syntaxe suivante :

  ```
  CAST(instr(string str, string substr) as INT)
  ```

  Sans cette étape, la vue sera considérée comme obsolète.
- length(string a) – Parce que le mappage d'Athena pour la fonction `length` renvoie BIGINT et non INT, utilisez la syntaxe suivante pour que la vue ne soit pas considérée comme obsolète :

  ```
  CAST(length(string str) as INT)
  ```

Fonctions de chaîne non prises en charge

Les fonctions de chaîne Hive suivantes ne sont pas prises en charge dans les vues Athena pour Hive.

```
ascii(string str)
character_length(string str)
decode(binary bin, string charset)
encode(string src, string charset)
elt(N int,str1 string,str2 string,str3 string,...)
field(val T,val1 T,val2 T,val3 T,...)
find_in_set(string str, string strList)
initcap(string A)
levenshtein(string A, string B)
locate(string substr, string str[, int pos])
octet_length(string str)
parse_url(string urlString, string partToExtract [, string keyToExtract])
printf(String format, Obj... args)
quote(String text)
regexp_extract(string subject, string pattern, int index)
repeat(string str, int n)
sentences(string str, string lang, string locale)
soundex(string A)
space(int n)
str_to_map(text[, delimiter1, delimiter2])
substring_index(string A, string delim, int count)
```

Fonctions XPath non prises en charge

Les fonctions XPath Hive telles que `xpath`, `xpath_short` et `xpath_int` ne sont pas pris en charge dans les vues Athena pour Hive.

Résolution des problèmes

Lorsque vous utilisez les vues Hive dans Athena, vous pouvez rencontrer les problèmes suivants :

- View ***<view name>*** is stale (La vue <nom de la vue> est obsolète) – Ce message indique généralement une incompatibilité de type entre la vue dans Hive et Athena. Si la même fonction présente dans la documentation des fonctions et opérateurs Hive LanguageManual UDF et Presto possède des signatures différentes, essayez de convertir le type de données incompatible.
- Fonction non enregistrée – Athena ne prend pas actuellement en charge cette fonction. Pour obtenir des détails, consultez les informations plus haut dans ce document.

Utilisation des AWS CLI métastores with Hive

Vous pouvez utiliser les commandes CLI `aws athena` pour gérer les catalogues de données de métastore Hive que vous utilisez avec Athena. Une fois que vous avez défini un ou plusieurs catalogues à utiliser avec Athena, vous pouvez les référencer dans vos commandes DDL et DML `aws athena`.

Utilisation du AWS CLI pour gérer les catalogues de métastores Hive

Enregistrer un catalogue : C reate-data-catalog

Pour enregistrer un catalogue de données, utilisez la commande `create-data-catalog`. Utilisez le paramètre `name` pour spécifier le nom que vous souhaitez utiliser pour le catalogue. Passage de l'ARN de la fonction Lambda à l'option `metadata-function` de l'argument `parameters`. Pour créer des identifications pour le nouveau catalogue, utilisez le paramètre `tags` avec une ou plusieurs paires d'arguments `Key=`*`key`*`,Value=`*`value`* séparées par des espaces.

L'exemple suivant enregistre le catalogue de métastore Hive nommé `hms-catalog-1`. La commande a été formatée pour être lisible.

```
$ aws athena create-data-catalog
 --name "hms-catalog-1"
 --type "HIVE"
 --description "Hive Catalog 1"
 --parameters "metadata-function=arn:aws:lambda:us-
east-1:111122223333:function:external-hms-service-v3,sdk-version=1.0"
 --tags Key=MyKey,Value=MyValue
 --region us-east-1
```

Afficher les détails du catalogue : G et-data-catalog

Pour afficher les détails d'un catalogue, passez le nom du catalogue à la commande `get-data-catalog`, comme dans l'exemple suivant.

```
$ aws athena get-data-catalog --name "hms-catalog-1" --region us-east-1
```

L'exemple de résultat suivant est au format JSON.

```
{
    "DataCatalog": {
        "Name": "hms-catalog-1",
```

```
        "Description": "Hive Catalog 1",
        "Type": "HIVE",
        "Parameters": {
            "metadata-function": "arn:aws:lambda:us-
east-1:111122223333:function:external-hms-service-v3",
            "sdk-version": "1.0"
        }
    }
}
```

Liste des catalogues enregistrés : L ist-data-catalogs

Pour répertorier les catalogues enregistrés, utilisez la commande `list-data-catalogs` et spécifiez éventuellement une région, comme dans l'exemple suivant. Les catalogues répertoriés incluent toujours AWS Glue.

```
$ aws athena list-data-catalogs --region us-east-1
```

L'exemple de résultat suivant est au format JSON.

```
{
    "DataCatalogs": [
        {
            "CatalogName": "AwsDataCatalog",
            "Type": "GLUE"
        },
        {
            "CatalogName": "hms-catalog-1",
            "Type": "HIVE",
            "Parameters": {
                "metadata-function": "arn:aws:lambda:us-
east-1:111122223333:function:external-hms-service-v3",
                "sdk-version": "1.0"
            }
        }
    ]
}
```

Mettre à jour un catalogue : U pdate-data-catalog

Pour mettre à jour un catalogue de données, utilisez la commande `update-data-catalog`, comme dans l'exemple suivant. La commande a été formatée pour être lisible.

```
$ aws athena update-data-catalog
 --name "hms-catalog-1"
 --type "HIVE"
 --description "My New Hive Catalog Description"
 --parameters "metadata-function=arn:aws:lambda:us-
east-1:111122223333:function:external-hms-service-new,sdk-version=1.0"
 --region us-east-1
```

Supprimer un catalogue : D elete-data-catalog

Pour supprimer un catalogue de données, utilisez la commande `delete-data-catalog`, comme dans l'exemple suivant.

```
$ aws athena delete-data-catalog --name "hms-catalog-1" --region us-east-1
```

Afficher les détails de la base de données : Get-database

Pour afficher les détails d'une base de données, passez le nom du catalogue et de la base de données à la commande `get-database`, comme dans l'exemple suivant.

```
$ aws athena get-database --catalog-name hms-catalog-1 --database-name mydb
```

L'exemple de résultat suivant est au format JSON.

```
{
    "Database": {
        "Name": "mydb",
        "Description": "My database",
        "Parameters": {
            "CreatedBy": "Athena",
            "EXTERNAL": "TRUE"
        }
    }
}
```

Liste des bases de données dans un catalogue : List-databases

Pour répertorier les bases de données dans un catalogue, utilisez la commande `list-databases` et spécifiez éventuellement une région, comme dans l'exemple suivant.

```
$ aws athena list-databases --catalog-name AwsDataCatalog --region us-west-2
```

L'exemple de résultat suivant est au format JSON.

```
{
    "DatabaseList": [
        {
            "Name": "default"
        },
        {
            "Name": "mycrawlerdatabase"
        },
        {
            "Name": "mydatabase"
        },
        {
            "Name": "sampledb",
            "Description": "Sample database",
            "Parameters": {
                "CreatedBy": "Athena",
                "EXTERNAL": "TRUE"
            }
        },
        {
            "Name": "tpch100"
        }
    ]
}
```

Affichage des détails du tableau : G et-table-metadata

Pour afficher les métadonnées d'une table, y compris les noms de colonnes et les types de données, transmettez le nom du catalogue, de la base de données et de la table à la commande `get-table-metadata`, comme dans l'exemple suivant.

```
$ aws athena get-table-metadata --catalog-name AwsDataCatalog --database-name mydb --
table-name cityuseragent
```

L'exemple de résultat suivant est au format JSON.

```
{
```

```
    "TableMetadata": {
        "Name": "cityuseragent",
            "CreateTime": 1586451276.0,
            "LastAccessTime": 0.0,
            "TableType": "EXTERNAL_TABLE",
            "Columns": [
                {
                    "Name": "city",
                    "Type": "string"
                },
                {
                    "Name": "useragent1",
                    "Type": "string"
                }
            ],
            "PartitionKeys": [],
            "Parameters": {
                "COLUMN_STATS_ACCURATE": "false",
                "EXTERNAL": "TRUE",
                "inputformat": "org.apache.hadoop.mapred.TextInputFormat",
                "last_modified_by": "hadoop",
                "last_modified_time": "1586454879",
                "location": "s3://DOC-EXAMPLE-BUCKET/",
                "numFiles": "1",
                "numRows": "-1",
                "outputformat":
 "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
                "rawDataSize": "-1",
                "serde.param.serialization.format": "1",
                "serde.serialization.lib":
 "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
                "totalSize": "61"
            }
        }
}
```

Affichage des métadonnées pour toutes les tables d'une base de données : L ist-table-metadata

Pour afficher les métadonnées de toutes les tables d'une base de données, transmettez le nom du catalogue et celui de la base de données à la commande `list-table-metadata`. La commande `list-table-metadata` est similaire à la commande `get-table-metadata`, sauf que vous ne spécifiez pas de nom de table. Pour limiter le nombre de résultats, vous pouvez utiliser l'option `--max-results`, comme dans l'exemple suivant.

```
$ aws athena list-table-metadata --catalog-name AwsDataCatalog --database-name sampledb
 --region us-east-1 --max-results 2
```

L'exemple de résultat suivant est au format JSON.

```
{
    "TableMetadataList": [
        {
            "Name": "cityuseragent",
            "CreateTime": 1586451276.0,
            "LastAccessTime": 0.0,
            "TableType": "EXTERNAL_TABLE",
            "Columns": [
                {
                    "Name": "city",
                    "Type": "string"
                },
                {
                    "Name": "useragent1",
                    "Type": "string"
                }
            ],
            "PartitionKeys": [],
            "Parameters": {
                "COLUMN_STATS_ACCURATE": "false",
                "EXTERNAL": "TRUE",
                "inputformat": "org.apache.hadoop.mapred.TextInputFormat",
                "last_modified_by": "hadoop",
                "last_modified_time": "1586454879",
                "location": "s3://DOC-EXAMPLE-BUCKET/",
                "numFiles": "1",
                "numRows": "-1",
                "outputformat":
 "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
                "rawDataSize": "-1",
                "serde.param.serialization.format": "1",
                "serde.serialization.lib":
 "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
                "totalSize": "61"
            }
        },
        {
            "Name": "clearinghouse_data",
```

```
            "CreateTime": 1589255544.0,
            "LastAccessTime": 0.0,
            "TableType": "EXTERNAL_TABLE",
            "Columns": [
                {
                    "Name": "location",
                    "Type": "string"
                },
                {
                    "Name": "stock_count",
                    "Type": "int"
                },
                {
                    "Name": "quantity_shipped",
                    "Type": "int"
                }
            ],
            "PartitionKeys": [],
            "Parameters": {
                "EXTERNAL": "TRUE",
                "inputformat": "org.apache.hadoop.mapred.TextInputFormat",
                "location": "s3://DOC-EXAMPLE-BUCKET/",
                "outputformat":
 "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
                "serde.param.serialization.format": "1",
                "serde.serialization.lib":
 "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
                "transient_lastDdlTime": "1589255544"
            }
        }
    ],
    "NextToken":
 "eyJsYXN0RXZhbHVhdGVkS2V5Ijp7IkhBU0hfS0VZIjp7InMiOiJ0Ljk0YWZjYjk1MjJjNTQ1YmU4Y2I5OWE5NTg0MjFjY
}
```

Exécution d'instructions DDL et DML

Lorsque vous utilisez les instructions DDL et DML AWS CLI pour exécuter, vous pouvez transmettre le nom du catalogue de métastores Hive de deux manières :

- Directement dans les déclarations qui le soutiennent.
- Au paramètre `--query-execution-context Catalog`.

Instructions DDL

L'exemple suivant transmet le nom du catalogue directement dans le cadre de l'instruction `show create table` DDL. La commande a été formatée pour être lisible.

```
$ aws athena start-query-execution
 --query-string "show create table hms-catalog-1.hms_tpch_partitioned.lineitem"
 --result-configuration "OutputLocation=s3://DOC-EXAMPLE-BUCKET/lambda/results"
```

L'exemple suivant d'instruction `show create table` DDL utilise le paramètre `Catalog` de `--query-execution-context` pour transmettre le nom du catalogue de métastore Hive `hms-catalog-1`. La commande a été formatée pour être lisible.

```
$ aws athena start-query-execution
 --query-string "show create table lineitem"
 --query-execution-context "Catalog=hms-catalog-1,Database=hms_tpch_partitioned"
 --result-configuration "OutputLocation=s3://DOC-EXAMPLE-BUCKET/lambda/results"
```

Instructions DML

L'exemple suivant d'instruction `select` DML transmet directement le nom du catalogue dans la requête. La commande a été formatée pour être lisible.

```
$ aws athena start-query-execution
 --query-string "select * from hms-catalog-1.hms_tpch_partitioned.customer limit 100"
 --result-configuration "OutputLocation=s3://DOC-EXAMPLE-BUCKET/lambda/results"
```

L'exemple suivant d'instruction `select` DML utilise le paramètre `Catalog` de `--query-execution-context` pour transmettre le nom du catalogue de métastore Hive `hms-catalog-1`. La commande a été formatée pour être lisible.

```
$ aws athena start-query-execution
 --query-string "select * from customer limit 100"
 --query-execution-context "Catalog=hms-catalog-1,Database=hms_tpch_partitioned"
 --result-configuration "OutputLocation=s3://DOC-EXAMPLE-BUCKET/lambda/results"
```

Implémentation de référence

Athena fournit une implémentation de référence de son connecteur pour le métastore Hive externe sur .com à l'adresse https://github.com/awslabs/. GitHub aws-athena-hive-metastore

L'implémentation de référence est un projet Apache Maven qui a les modules suivants :

- **`hms-service-api`** – Contient les opérations d'API entre la fonction Lambda et les clients du service Athena. Ces opérations d'API sont définies dans l'interface `HiveMetaStoreService`. Comme il s'agit d'un contrat de service, vous ne devriez rien changer dans ce module.
- **`hms-lambda-handler`** – Un ensemble de gestionnaires Lambda par défaut qui traitent tous les appels de l'API de métastore Hive. La classe `MetadataHandler` est le répartiteur pour tous les appels d'API. Vous n'avez pas besoin de modifier ce package.
- **`hms-lambda-func`** – Exemple de fonction Lambda comportant les composants suivants.
  - **`HiveMetaStoreLambdaFunc`** – Exemple de fonction Lambda qui étend `MetadataHandler`.
  - **`ThriftHiveMetaStoreClient`** – Client Thrift qui communique avec le métastore Hive. Ce client est écrit pour Hive 2.3.0. Si vous utilisez une version différente de Hive, vous devrez peut-être mettre à jour cette classe pour vous assurer que les objets de réponse sont compatibles.
  - **`ThriftHiveMetaStoreClientFactory`** – Contrôle le comportement de la fonction Lambda. Par exemple, vous pouvez fournir votre propre ensemble de fournisseurs de gestionnaires en remplaçant la méthode `getHandlerProvider()`.
  - `hms.properties` – Configure la fonction Lambda. La plupart des cas nécessitent la mise à jour des deux propriétés suivantes uniquement.
    - `hive.metastore.uris` – l'URI du métastore Hive au format `thrift://<host_name>:9083`.
    - `hive.metastore.response.spill.location` : l'emplacement Amazon S3 pour stocker les objets de réponse lorsque leur taille dépasse un seuil donné (par exemple, 4 Mo). Le seuil est défini dans la propriété `hive.metastore.response.spill.threshold`. Il n'est pas recommandé de modifier la valeur par défaut.

> **Note**
>
> Ces deux propriétés peuvent être remplacées par les variables d'environnement Lambda `HMS_URIS` et `SPILL_LOCATION`. Utilisez ces variables au lieu de recompiler le code source de la fonction Lambda lorsque vous souhaitez utiliser la fonction avec un métastore Hive ou un emplacement de déversement différent.

- **`hms-lambda-layer`** – Un projet d'assemblage Maven qui place `hms-service-api`, `hms-lambda-handler` et leurs dépendances dans un fichier `.zip`. Le fichier `.zip` est enregistré en tant que couche Lambda pour une utilisation par plusieurs fonctions Lambda.

- **`hms-lambda-rnp`** – Enregistre les réponses d'une fonction Lambda, puis les utilise pour rejouer la réponse. Vous pouvez utiliser ce modèle pour simuler des réponses Lambda à des fins de test.

Construire les artefacts vous-même

La plupart des cas d'utilisation n'exigent pas que vous modifiiez l'implémentation de référence. Toutefois, si nécessaire, vous pouvez modifier le code source, créer les artefacts vous-même et les télécharger dans un emplacement Simple Storage Service (Amazon S3).

Avant de créer les artefacts, mettez à jour les propriétés `hive.metastore.uris` et `hive.metastore.response.spill.location` dans le fichier `hms.properties` du module `hms-lambda-func`.

Pour construire les artefacts, vous devez avoir Apache Maven installé et exécuter la commande `mvn install`. Cela génère le fichier `.zip` de couche dans le dossier de sortie appelé `target` dans le module `hms-lambda-layer` et le fichier `.jar` de fonction Lambda dans le module `hms-lambd-func`.

Utilisation de la requête fédérée d'Amazon Athena

Si vos données proviennent d'autres sources que Simple Storage Service (Amazon S3), vous pouvez utiliser la requête fédérée d'Athena pour interroger les données sur place ou créer des pipelines qui extraient les données de plusieurs sources et les stockent dans Simple Storage Service (Amazon S3). La requête fédérée d'Athena vous permet d'exécuter des requêtes SQL sur des données stockées dans des sources de données relationnelles, non relationnelles, objets et personnalisées.

Athena utilise des connecteurs de source de données qui s'exécutent AWS Lambda pour exécuter des requêtes fédérées. Un connecteur de source de données est un morceau de code qui peut traduire des données entre votre source de données cible et Athena. Vous pouvez vous représenter un connecteur comme une extension du moteur de requêtes Athena. Des connecteurs de source de données Athena prédéfinis existent pour les sources de données telles qu'Amazon Logs, CloudWatch Amazon DynamoDB, Amazon DocumentDB et Amazon RDS, ainsi que pour les sources de données relationnelles compatibles JDBC telles que MySQL et PostgreSQL sous licence Apache 2.0. Vous pouvez également utiliser le kit Athena Query Federation SDK pour écrire des connecteurs personnalisés. Pour choisir, configurer et déployer un connecteur de source de données sur votre compte, vous pouvez utiliser les consoles Athena et Lambda ou le AWS Serverless Application Repository. Une fois que vous avez déployé des connecteurs de source de données, le connecteur est associé à un catalogue que vous pouvez spécifier dans les requêtes SQL. Vous pouvez combiner

des instructions SQL de plusieurs catalogues et couvrir plusieurs sources de données à l'aide d'une seule requête.

Lorsqu'une requête est envoyée à une source de données, Athena invoque le connecteur correspondant pour identifier les parties des tables qui doivent être lues, gère le parallélisme et réduit les prédicats de filtre. En fonction de l'utilisateur qui soumet la requête, les connecteurs peuvent autoriser ou restreindre l'accès à des éléments de données spécifiques. Les connecteurs utilisent Apache Arrow comme format pour renvoyer les données demandées dans une requête, ce qui permet d'implémenter des connecteurs dans des langages tels que C, C++, Java, Python et Rust. Étant donné que les connecteurs sont traités dans Lambda, ils peuvent être utilisés pour accéder aux données de n'importe quelle source de données du cloud ou sur site accessible depuis Lambda.

Pour écrire votre propre connecteur de source de données, vous pouvez utiliser le kit Athena Query Federation SDK pour personnaliser l'un des connecteurs prédéfinis fournis et gérés par Amazon Athena. Vous pouvez modifier une copie du code source depuis le GitHub référentiel, puis utiliser l'outil de publication Connector pour créer votre propre AWS Serverless Application Repository package.

> **Note**
>
> Il est possible que des développeurs tiers aient utilisé le kit Athena Query Federation SDK pour écrire des connecteurs de sources de données. Pour tout problème de support ou de licence concernant ces connecteurs de sources de données, veuillez vous adresser à votre fournisseur de connecteurs. Ces connecteurs ne sont ni testés ni pris en charge par AWS.

Pour une liste des connecteurs de sources de données écrits et testés par Athena, voir Connecteurs de source de données disponibles.

Pour plus d'informations sur l'écriture de votre propre connecteur de source de données, voir Exemple de connecteur Athena activé. GitHub

Considérations et restrictions

- Versions du moteur : la requête fédérée d'Athena n'est prise en charge que par la version 2 du moteur Athena et les versions ultérieures. Pour plus d'informations sur les versions du moteur Athena, voir Gestion des versions du moteur Athena.

- Vues : vous pouvez désormais créer et interroger des vues sur des sources de données fédérées. Les vues fédérées sont stockées dans la source de données sous-jacente AWS Glue, et non dans celle-ci. Pour plus d'informations, consultez Interrogation des vues fédérées.
- Opérations d'écriture : les opérations d'écriture telles que INSERT INTO ne sont pas pris en charge. Si vous tentez de le faire, le message d'erreur suivant peut s'afficher : This operation is currently not supported for external catalogs (Cette opération n'est actuellement pas prise en charge pour les catalogues externes).
- Tarification : pour des informations sur la tarification, consultez la rubrique Tarification Amazon Athena.

  Pilote JDBC : pour utiliser le pilote JDBC avec des requêtes fédérées ou un métastore Hive externe, incluez `MetadataRetrievalMethod=ProxyAPI` dans votre chaîne de connexion JDBC. Pour plus d'informations sur le pilote JDBC, voir Connexion à Amazon Athena avec JDBC.
- Secrets Manager : pour utiliser la fonction de requête fédérée d'Athena avec AWS Secrets Manager, vous devez configurer un point de terminaison privé Amazon VPC pour Secrets Manager. Pour plus d'informations, consultez la rubrique Création d'un point de terminaison privé VPC Secrets Manager du Guide de l'utilisateur AWS Secrets Manager .

Les connecteurs de source de données peuvent nécessiter l'accès aux ressources suivantes pour fonctionner correctement. Si vous utilisez un connecteur prédéfini, vérifiez les informations relatives au connecteur pour vous assurer que vous avez correctement configuré votre VPC. Assurez-vous également que les principals IAM exécutant des requêtes et créant des connecteurs disposent de privilèges sur les actions requises. Pour plus d'informations, consultez Exemple de politiques d'autorisation IAM pour autoriser la requête fédérée Athena.

- Simple Storage Service (Amazon S3) : outre l'écriture des résultats des requêtes dans l'emplacement des résultats des requêtes Athena dans Simple Storage Service (Amazon S3), les connecteurs de données écrivent également dans un compartiment de déversement dans Simple Storage Service (Amazon S3). Une connectivité et des autorisations d'accès à cet emplacement Simple Storage Service (Amazon S3) sont requises.
- Athena : les sources de données ont besoin d'une connectivité vers Athena et vice versa pour vérifier l'état des requêtes et empêcher le surbalayage.
- AWS Glue Data Catalog : la connectivité et des autorisations sont nécessaires si votre connecteur utilise le catalogue de données pour les métadonnées supplémentaires ou principales.

Vidéos

Regardez les vidéos suivantes pour en savoir plus sur l'utilisation de la requête fédérée d'Athena.

Vidéo : Analyser les résultats d'une requête fédérée dans Amazon Athena sur Amazon QuickSight

La vidéo suivante montre comment analyser les résultats d'une requête fédérée Athena dans Amazon. QuickSight

Analyser les résultats d'une requête fédérée dans Amazon Athena sur Amazon QuickSight

Vidéo : Gaming Analytics Pipeline

La vidéo suivante montre comment déployer un pipeline de données (Data Pipeline) évolutif sans serveur pour intégrer, stocker et analyser des données de télémétrie provenant de jeux et de services à l'aide de requêtes fédérées Amazon Athena.

Pipeline d'analytique de jeu

Connecteurs de source de données disponibles

Cette section répertorie les connecteurs de source de données Athena préconstruits que vous pouvez utiliser pour interroger une variété de sources de données externes à Simple Storage Service (Amazon S3). Pour utiliser un connecteur dans vos requêtes Athena, configurez-le et déployez-le sur votre compte.

Considérations et restrictions

- Certains connecteurs prédéfinis nécessitent de créer un VPC et un groupe de sécurité avant de pouvoir utiliser le connecteur. Pour plus d'informations sur la création de VPC, consultez Création d'un VPC pour un connecteur de source de données.
- Pour utiliser la fonctionnalité Athena Federated Query avec AWS Secrets Manager, vous devez configurer un point de terminaison privé Amazon VPC pour Secrets Manager. Pour plus d'informations, consultez la rubrique Création d'un point de terminaison privé VPC Secrets Manager du Guide de l'utilisateur AWS Secrets Manager .
- L'exécution des requêtes incluant un prédicat prend beaucoup plus de temps sur les connecteurs qui ne prennent pas en charge la poussée vers le bas des prédicats. Pour les petits jeux de données, très peu de données sont analysées et les requêtes prennent environ deux minutes en moyenne. Toutefois, pour les jeux de données volumineux, de nombreuses requêtes peuvent expirer.

- Certaines sources de données fédérées utilisent une terminologie différente de celle d'Athena pour désigner des objets de données. Pour plus d'informations, consultez Athena et qualificateurs de noms de tables fédérées.
- Pour les connecteurs qui ne prennent pas en charge la pagination lorsque vous répertoriez des tables, le service web peut expirer si votre base de données comporte de nombreuses tables et métadonnées. Les connecteurs suivants fournissent une prise en charge de la pagination pour répertorier les tables :
  - DocumentDB
  - DynamoDB
  - MySQL
  - OpenSearch
  - Oracle
  - PostgreSQL
  - Redshift
  - SQL Server

Informations supplémentaires

- Pour plus d'informations sur le déploiement d'un connecteur de source de données Athena, voir Déploiement d'un connecteur de source de données.
- Pour plus d'informations sur les requêtes qui utilisent les connecteurs de source de données Athena, consultez Exécution de requêtes fédérées.
- Pour des informations détaillées sur les connecteurs de source de données Athena, voir Connecteurs disponibles sur GitHub.

Connecteurs de source de données Athena

- Connecteur Amazon Athena pour Azure Data Lake Storage (ADLS) Gen2
- Connecteur Amazon Athena pour Azure Synapse
- Connecteur Amazon Athena pour Cloudera Hive
- Connecteur Amazon Athena pour Cloudera Impala
- Connecteur Amazon Athena CloudWatch
- Connecteur Amazon Athena Metrics CloudWatch
- Connecteur CMDB Amazon Athena AWS

- Connecteur Amazon Athena pour Db2 IBM
- Connecteur Amazon Athena IBM Db2 AS/400 (Db2 iSeries)
- Connecteur Amazon Athena pour DocumentDB
- Connecteur Amazon Athena pour DynamoDB
- Connecteur Amazon Athena pour Google BigQuery
- Connecteur Amazon Athena Google Cloud Storage
- Connecteur Amazon Athena pour HBase
- Connecteur Amazon Athena pour Hortonworks
- Connecteur Amazon Athena Apache Kafka
- Connecteur Amazon Athena pour MSK
- Connecteur Amazon Athena pour MySQL
- Connecteur Amazon Athena pour Neptune
- Connecteur Amazon Athena OpenSearch
- Connecteur Amazon Athena pour Oracle
- Connecteur Amazon Athena pour PostgreSQL
- Connecteur Amazon Athena pour Redis
- Connecteur Amazon Athena pour Redshift
- Connecteur Amazon Athena pour SAP HANA
- Connecteur Amazon Athena pour Snowflake
- Connecteur Amazon Athena pour Microsoft SQL Server
- Connecteur Amazon Athena pour Teradata
- Connecteur Amazon Athena pour Timestream
- Connecteur Amazon Athena pour TPC Benchmark DS (TPC-DS)
- Connecteur Amazon Athena pour Vertica

> **Note**
>
> La AthenaJdbcConnector(dernière version 2022.4.1) est obsolète. Utilisez plutôt un connecteur spécifique à la base de données, comme ceux de MySQL, Redshift ou PostgreSQL.

Connecteur Amazon Athena pour Azure Data Lake Storage (ADLS) Gen2

Le connecteur Amazon Athena pour Azure Data Lake Storage (ADLS) Gen2 permet à Amazon Athena d'exécuter des requêtes SQL sur des données stockées sur ADLS. Athena ne peut pas accéder directement aux fichiers stockés dans le lac de données.

- Flux de travail : le connecteur implémente l'interface JDBC, qui utilise le pilote `com.microsoft.sqlserver.jdbc.SQLServerDriver`. Le connecteur transmet les requêtes au moteur Azure Synapse, qui accède ensuite au lac de données.
- Gestion des données et S3 : normalement, le connecteur Lambda interroge les données directement sans les transférer vers Amazon S3. Toutefois, lorsque les données renvoyées par la fonction Lambda dépassent les limites Lambda, elles sont écrites dans le compartiment de déversement Amazon S3 que vous spécifiez afin qu'Athena puisse lire l'excédent.
- Authentification AAD : AAD peut être utilisé comme méthode d'authentification pour le connecteur Azure Synapse. Pour utiliser AAD, la chaîne de connexion JDBC utilisée par le connecteur doit contenir les paramètres d'URL `authentication=ActiveDirectoryServicePrincipal`, `AADSecurePrincipalId` et `AADSecurePrincipalSecret`. Ces paramètres peuvent y être transmis directement ou par Secrets Manager.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Les types de données de date et d'horodatage dans des conditions de filtre doivent être convertis en types de données appropriés.

Conditions

Les termes suivants se rapportent au connecteur Azure Data Lake Storage Gen2.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Azure Data Lake Storage Gen2.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
datalakegentwo://${jdbc_connection_string}
```

Utilisation d’un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l’aide d’une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`DataLakeGen2MuxCompositeHandler`
Gestionnaire de métadonnées	`DataLakeGen2MuxMetadataHandler`
Gestionnaire d’enregis trements	`DataLakeGen2MuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
$*catalog*_connecti on_string	Obligatoire. Chaîne de connexion d’instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d’Athena est `mydatalakegentwocatalog` , le nom de la variable d’environnement est alors `mydatalakegentwoca talog_connection_string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *AWS_LAMBD A_FUNCTION_NAME* `}`.

Les exemples de propriétés suivants concernent une fonction Lambda MUX à DataLakeGen 2 niveaux qui prend en charge deux instances de base de données `datalakegentwo1` : (par défaut) et. `datalakegentwo2`

Propriété	Valeur
`default`	`datalakegentwo://jdbc:sqlserver://adlsgentwo1. hostname:port;databaseName= database_name ; ${secret1_name }`
`datalakegentwo_catalog1_connection_string`	`datalakegentwo://jdbc:sqlserver://adlsgentwo1. hostname:port;databaseName= database_name ; ${secret1_name }`
`datalakegentwo_catalog2_connection_string`	`datalakegentwo://jdbc:sqlserver://adlsgentwo2. hostname:port;databaseName= database_name ; ${secret2_name }`

Fourniture des informations d'identification

Pour fournir un nom d'utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${secret1_name}`.

```
datalakegentwo://jdbc:sqlserver://hostname:port;databaseName=database_name;
${secret1_name}
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
datalakegentwo://
jdbc:sqlserver://
hostname:port;databaseName=database_name;user=user_name;password=password
```

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance Azure Data Lake Storage Gen2.

Type de gestionnaire	Classe
Gestionnaire de composites	`DataLakeGen2CompositeHandler`
Gestionnaire de métadonnées	`DataLakeGen2MetadataHandler`
Gestionnaire d'enregistrements	`DataLakeGen2RecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une instance Azure Data Lake Storage Gen2 unique prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`datalakegentwo://jdbc:sqlserver://` *`hostname`*`:`*`port`*`;databaseName=;${` *`secret_name`* `}`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant indique les types de données correspondants pour ADLS Gen2 et Arrow.

ADLS Gen2	Flèche
bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT
int	INT
bigint	BIGINT
decimal	DECIMAL
numeric	FLOAT8
smallmoney	FLOAT8
money	DECIMAL
float[24]	FLOAT4
float[53]	FLOAT8
real	FLOAT4
datetime	Date(MILLISECOND)
datetime2	Date(MILLISECOND)
smalldatetime	Date(MILLISECOND)
date	Date(DAY)
time	VARCHAR
datetimeoffset	Date(MILLISECOND)

ADLS Gen2	Flèche
char[n]	VARCHAR
varchar[n/max]	VARCHAR

Partitions et déversements

Azure Data Lake Storage Gen2 utilise un stockage blob Gen2 compatible avec Hadoop pour stocker des fichiers de données. Les données de ces fichiers sont interrogées à partir du moteur Azure Synapse. Le moteur Azure Synapse traite les données Gen2 stockées dans les systèmes de fichiers comme des tables externes. Les partitions sont mises en œuvre en fonction du type de données. Si les données ont déjà été partitionnées et distribuées au sein du système de stockage Gen2, le connecteur extrait les données sous forme de division unique.

Performance

Le connecteur Azure Data Lake Storage Gen2 présente des performances plus lentes lors de l'exécution de plusieurs requêtes à la fois et fait l'objet d'une limitation.

Le connecteur Athena Azure Data Lake Storage Gen2 effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Des prédicats simples et des expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution de la requête.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena Azure Data Lake Storage Gen2 peut combiner ces expressions et les pousser directement vers Azure Data Lake Storage Gen2 pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena Azure Data Lake Storage Gen2 suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN_OR_EQUAL, NULL_IF, IS_NULL

- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%');
```

Requêtes directes

Le connecteur Azure Data Lake Storage Gen2 prend en charge les requêtes passthrough. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes passthrough avec Azure Data Lake Storage Gen2, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Azure Data Lake Storage Gen2. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les dernières informations sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur Azure Data Lake Storage Gen2 sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Azure Synapse

Le connecteur Amazon Athena pour Azure Synapse Analytics permet à Amazon Athena d'exécuter des requêtes SQL sur vos bases de données Azure Synapse à l'aide de JDBC.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Dans des conditions de filtre, vous devez lancer les types de données `DateetTimestamp` vers le type de données approprié.
- Pour rechercher des valeurs négatives de type `Real` et `Float`, utilisez l'opérateur <= ou >=.
- Les types de données `binary`, `varbinary`, `image` et `rowversion` ne sont pas pris en charge.

Conditions

Les termes suivants se rapportent au connecteur Synapse.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Synapse.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
synapse://${jdbc_connection_string}
```

Utilisation d’un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l’aide d’une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`SynapseMuxCompositeHandler`
Gestionnaire de métadonnées	`SynapseMuxMetadataHandler`
Gestionnaire d’enregis trements	`SynapseMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
$*catalog*_connecti on_string	Obligatoire. Chaîne de connexion d’instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d’Athena est `mysynapsecatalog` , le nom de la variable d’environnement est alors `mysynapsecatalog_connection _string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *AWS_LAMBD A_FUNCTION_NAME* `}`.

Les exemples de propriétés suivants concernent une fonction Synapse MUX Lambda qui prend en charge deux instances de base de données :`synapse1` (par défaut) et `synapse2`.

Propriété	Valeur
default	synapse://jdbc:synapse://synapse1.hostname:port;databaseName= *<database_name>* ;${*secret1_name* }
synapse_catalog1_connection_string	synapse://jdbc:synapse://synapse1.hostname:port;databaseName= *<database_name>* ;${*secret1_name* }
synapse_catalog2_connection_string	synapse://jdbc:synapse://synapse2.hostname:port;databaseName= *<database_name>* ;${*secret2_name* }

Fourniture des informations d'identification

Pour fournir un nom d'utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret $ {secret_name}.

```
synapse://jdbc:synapse://hostname:port;databaseName=<database_name>;${secret_name}
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
synapse://jdbc:synapse://
hostname:port;databaseName=<database_name>;user=<user>;password=<password>
```

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance Synapse.

Type de gestionnaire	Classe
Gestionnaire de composites	`SynapseCompositeHandler`
Gestionnaire de métadonnées	`SynapseMetadataHandler`
Gestionnaire d'enregis trements	`SynapseRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une instance Synapse unique prise en charge par une fonction Lambda.

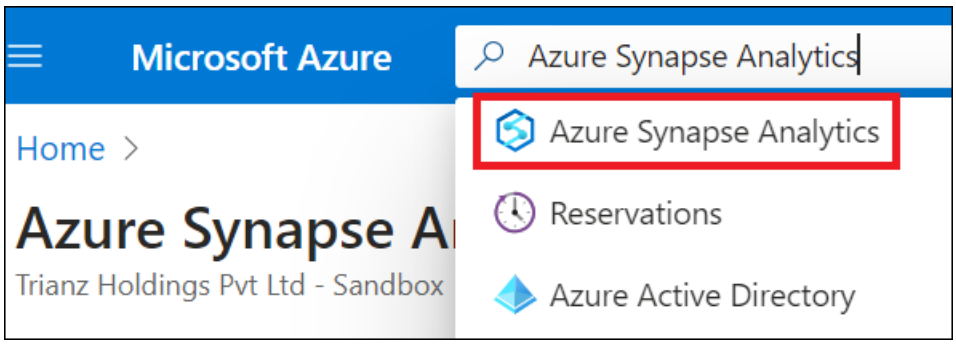
Propr	Valeur
`defa`	`synapse://jdbc:sqlserver://hostname:port;databaseName=` *`<database_name>`* `;${`*`secret_name`*`}`

Configuration de l'authentification Active Directory

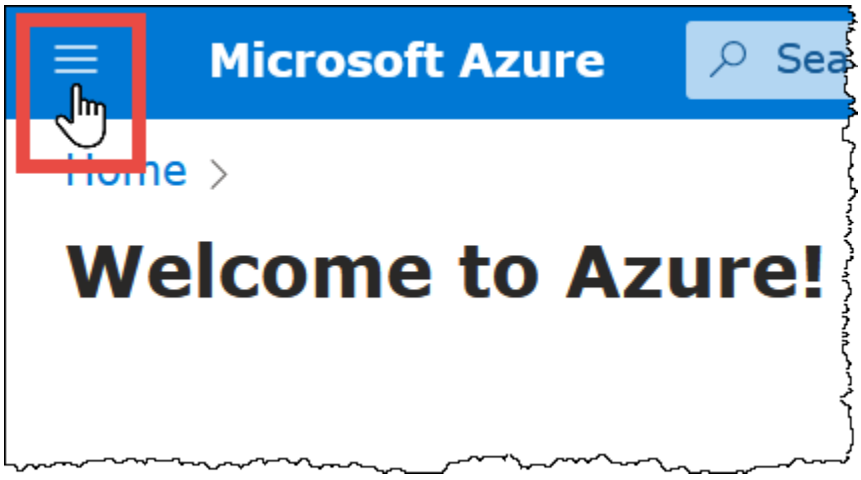
Le connecteur Amazon Athena Azure Synapse prend en charge l'authentification Microsoft Active Directory. Avant de commencer, vous devez configurer un utilisateur administratif sur le portail Microsoft Azure, puis l'utiliser AWS Secrets Manager pour créer un secret.

Pour configurer l'utilisateur administratif Active Directory

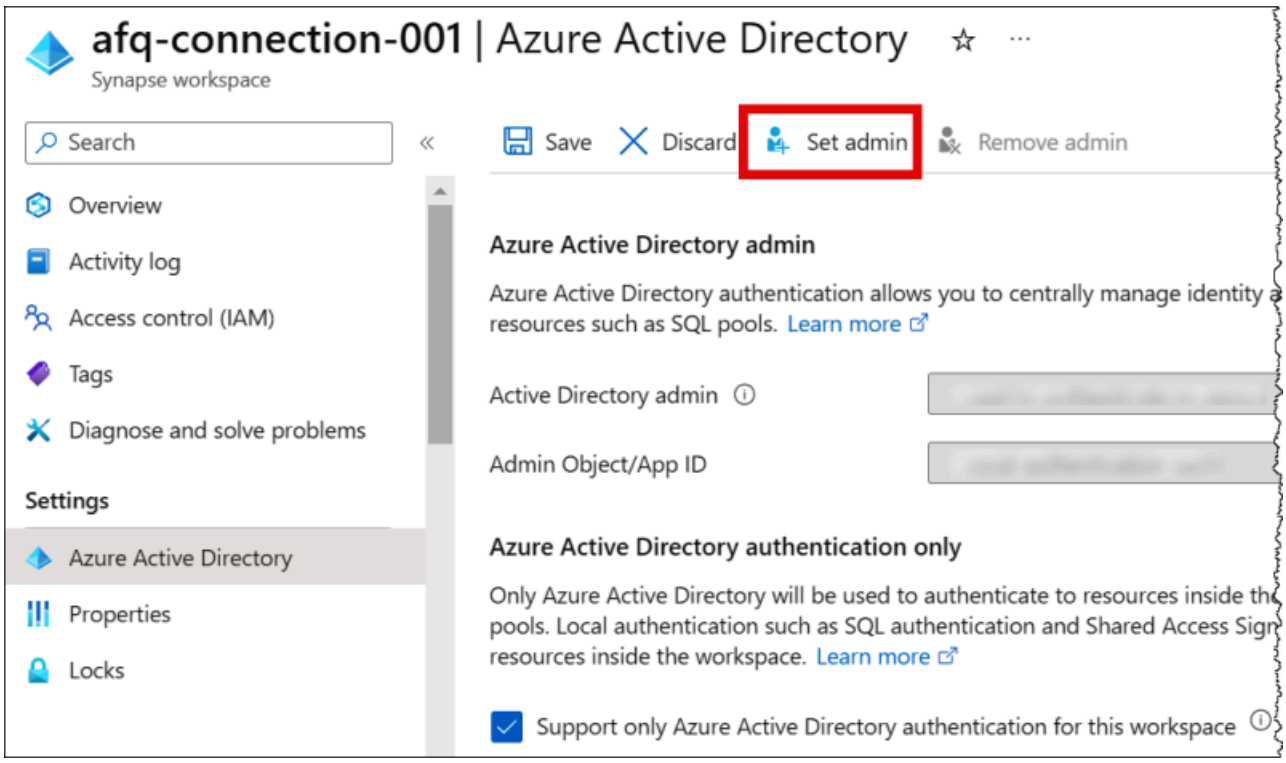
1. À l'aide d'un compte doté de privilèges administratifs, connectez-vous au portail Microsoft Azure à l'adresse https://portal.azure.com/.
2. Dans le champ de recherche, saisissez Azure Synapse Analytics, puis choisissez Azure Synapse Analytics.



3. Ouvrez le menu de gauche.



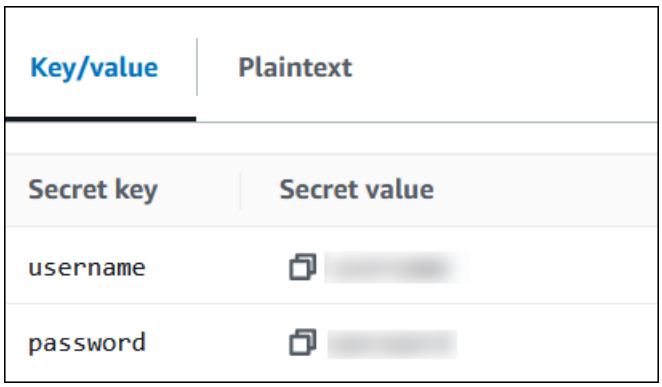
4. Dans le panneau de navigation, choisissez Azure Active Directory.
5. Dans l'onglet Définir l'administrateur, définissez l'administrateur Active Directory sur un utilisateur nouveau ou existant.



6. Dans AWS Secrets Manager, stockez le nom d'utilisateur et le mot de passe de l'administrateur. Pour plus d'informations sur la création d'un secret dans Secrets Manager, voir la rubrique Créer un secret AWS Secrets Manager.

Visualisation de votre secret dans Secrets Manager

1. Ouvrez la console Secrets Manager en suivant le lien https://console.aws.amazon.com/secretsmanager/.
2. Dans le volet de navigation, sélectionnez Secrets.
3. Sur la page Secrets, choisissez le lien vers votre secret.
4. Sur la page de détails de votre secret, sélectionnez Retrieve secret value (Récupérer la valeur du secret).



Modification de la chaîne de connexion

Pour activer l'authentification Active Directory pour le connecteur, modifiez la chaîne de connexion en utilisant la syntaxe suivante :

```
synapse://jdbc:synapse://
hostname:port;databaseName=database_name;authentication=ActiveDirectoryPassword;
{secret_name}
```

En utilisant ActiveDirectoryServicePrincipal

Le connecteur Amazon Athena Azure Synapse prend également en charge `ActiveDirectoryServicePrincipal`. Pour activer celui-ci, modifiez la chaîne de connexion comme suit.

```
synapse://jdbc:synapse://
hostname:port;databaseName=database_name;authentication=ActiveDirectoryServicePrincipal;
{secret_name}
```

Comme `secret_name`, spécifiez l'identifiant de l'application ou du client comme nom d'utilisateur et le secret de l'identité principale du service dans le mot de passe.

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.

Paramètre	Description
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant indique les types de données correspondants pour Synapse et Apache Arrow.

Synapse	Flèche
bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT
int	INT
bigint	BIGINT
decimal	DECIMAL
numeric	FLOAT8
smallmoney	FLOAT8
money	DECIMAL
float[24]	FLOAT4
float[53]	FLOAT8
real	FLOAT4

Synapse	Flèche
datetime	Date(MILLISECOND)
datetime2	Date(MILLISECOND)
smalldatetime	Date(MILLISECOND)
date	Date(DAY)
time	VARCHAR
datetimeoffset	Date(MILLISECOND)
char[n]	VARCHAR
varchar[n/max]	VARCHAR
nchar [n]	VARCHAR
nvarchar[n/max]	VARCHAR

Partitions et déversements

Une partition est représentée par une seule colonne de partition de type `varchar`. Synapse prend en charge le partitionnement par plages. Le partitionnement est donc implémenté en extrayant la colonne de partition et la plage de partitions des tables de métadonnées Synapse. Ces valeurs de plage sont utilisées pour créer les divisions.

Performance

La sélection d'un sous-ensemble de colonnes ralentit considérablement l'exécution de la requête. Le connecteur présente une limitation importante due à la simultanéité.

Le connecteur Athena Synapse effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Des prédicats simples et des expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution de la requête.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena Synapse peut combiner ces expressions et les pousser directement vers Synapse pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena Synapse suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN_OR_EQUAL, NULL_IF, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%');
```

Requêtes directes

Le connecteur Synapse prend en charge les requêtes passthrough. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Synapse, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
```

```
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Synapse. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

- Pour consulter un article expliquant comment utiliser Amazon QuickSight et Amazon Athena Federated Query pour créer des tableaux de bord et des visualisations à partir des données stockées dans les bases de données Microsoft Azure Synapse, consultez la section Effectuer des analyses multicloud à l'aide d'Amazon, QuickSight Amazon Athena Federated Query et Microsoft Azure Synapse sur le blog Big Data.AWS
- Pour obtenir les dernières informations sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur Synapse sur .com. GitHub
- Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Cloudera Hive

Le connecteur Amazon Athena pour Cloudera Hive permet à Athena d'exécuter des requêtes SQL sur la distribution Hadoop Cloudera Hive. Le connecteur transforme vos requêtes Athena SQL en syntaxe HiveQL équivalente.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l’aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d’informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

- Avant d'utiliser ce connecteur, configurez un VPC et un groupe de sécurité. Pour plus d'informations, consultez Création d'un VPC pour un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .

Conditions

Les termes suivants se rapportent au connecteur Cloudera Hive.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Cloudera Hive.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
hive://${jdbc_connection_string}
```

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`HiveMuxCompositeHandler`
Gestionnaire de métadonnées	`HiveMuxMetadataHandler`
Gestionnaire d'enregis trements	`HiveMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
$*catalog*_connecti on_string	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est myhivecatalog , le nom de la variable

Paramètre	Description
	d'environnement est alors `myhivecatalog_connection_string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${ AWS_LAMBDA_FUNCTION_NAME }`.

Les exemples de propriétés suivants concernent une fonction Hive MUX Lambda qui prend en charge deux instances de base de données :`hive1` (par défaut) et `hive2`.

Propriété	Valeur
`default`	`hive://jdbc:hive2://hive1:10000/default?${Test/RDS/hive1}`
`hive2_catalog1_connection_string`	`hive://jdbc:hive2://hive1:10000/default?${Test/RDS/hive1}`
`hive2_catalog2_connection_string`	`hive://jdbc:hive2://hive2:10000/default?UID=sample&PWD=sample`

Fourniture des informations d'identification

Pour fournir un nom d'utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

> **Important**
>
> Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets

Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

  Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${Test/RDS/hive1}`.

```
hive://jdbc:hive2://hive1:10000/default?...&${Test/RDS/hive1}&...
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
hive://jdbc:hive2://hive1:10000/default?...&UID=sample2&PWD=sample2&...
```

Actuellement, le connecteur Cloudera Hive reconnaît les propriétés JDBC `UID` et `PWD`.

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance Cloudera Hive.

Type de gestionnaire	Classe
Gestionnaire de composites	`HiveCompositeHandler`
Gestionnaire de métadonnées	`HiveMetadataHandler`
Gestionnaire d'enregistrements	`HiveRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une instance Cloudera Hive unique prise en charge par une fonction Lambda.

Propriété	Valeur
default	`hive://jdbc:hive2://hive1:10000/default?secret=${Test/RDS/hive1}`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant indique les types de données correspondants pour JDBC, Cloudera Hive et Arrow.

JDBC	Cloudera Hive	Flèche
Booléen	Booléen	Bit
Entier	TINYINT	Tiny
Court	SMALLINT	Smallint
Entier	INT	Int
Long	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8
Date	date	DateDay
Horodatage	timestamp	DateMilli
Chaîne	VARCHAR	Varchar

JDBC	Cloudera Hive	Flèche
Octets	octets	Varbinary
BigDecimal	Décimal	Décimal
ARRAY	N/A (voir la remarque)	Liste

> **Note**
>
> Hortonworks Cloudera Hive ne prend actuellement pas en charge les types d'agrégat `ARRAY`, `MAP`, `STRUCT` ou `UNIONTYPE`. Les colonnes de types d'agrégats sont traitées comme des colonnes `VARCHAR` dans SQL.

Partitions et déversements

Les partitions sont utilisées pour déterminer comment générer des divisions pour le connecteur. Athena construit une colonne synthétique de type `varchar` qui représente le schéma de partitionnement de la table afin d'aider le connecteur à générer des divisions. Le connecteur ne modifie pas la définition réelle de la table.

Performance

Cloudera Hive prend en charge les partitions statiques. Le connecteur Athena Cloudera Hive peut récupérer les données de ces partitions en parallèle. Si vous souhaitez interroger des jeux de données très volumineux avec une distribution de partition uniforme, le partitionnement statique est fortement recommandé. Le connecteur Cloudera Hive résiste à la limitation due à la simultanéité.

Le connecteur Athena Cloudera Hive effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Les clauses `LIMIT`, les prédicats simples et les expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution des requêtes.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena Cloudera Hive peut combiner ces expressions et les pousser directement vers Cloudera Hive pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena Cloudera Hive suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%')
LIMIT 10;
```

Requêtes directes

Le connecteur Cloudera Hive prend en charge les requêtes directes. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Cloudera Hive, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
```

```
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Cloudera Hive. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les dernières informations sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur Cloudera Hive sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Cloudera Impala

Le connecteur Amazon Athena Cloudera Impala permet à Athena d'exécuter des requêtes SQL sur la distribution Cloudera Impala. Le connecteur transforme vos requêtes SQL Athena en une syntaxe Impala équivalente.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l’aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d’informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.
- Avant d’utiliser ce connecteur, configurez un VPC et un groupe de sécurité. Pour plus d’informations, consultez Création d'un VPC pour un connecteur de source de données.

Limites

- Les opérations DDL d’écriture ne sont pas prises en charge.

- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .

Conditions

Les termes suivants se rapportent au connecteur Cloudera Impala.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d’enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d’environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d’environnement Lambda de cette section pour configurer le connecteur Cloudera Impala.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à un cluster Impala.

```
impala://${jdbc_connection_string}
```

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`ImpalaMuxCompositeHandler`
Gestionnaire de métadonnées	`ImpalaMuxMetadataHandler`
Gestionnaire d'enregis trements	`ImpalaMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
`$catalog_connecti on_string`	Obligatoire. Chaîne de connexion à un cluster Impala pour un catalogue Athena. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `myimpalacatalog` , le nom de la variable d'environnement est alors `myimpalac atalog_connection_string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${ AWS_LAMBD A_FUNCTION_NAME }`.

Les exemples de propriétés suivants concernent une fonction Impala MUX Lambda qui prend en charge deux instances de base de données :`impala1` (par défaut) et `impala2`.

Propriété	Valeur
`default`	`impala://jdbc:impala://some.impala.host.name:21050/?${Test/impala1}`
`impala_catalog1_connection_string`	`impala://jdbc:impala://someother.impala.host.name:21050/?${Test/impala1}`
`impala_catalog2_connection_string`	`impala://jdbc:impala://another.impala.host.name:21050/?UID=sample&PWD=sample`

Fourniture des informations d'identification

Pour fournir un nom d'utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${Test/impala1host}`.

```
impala://jdbc:impala://Impala1host:21050/?...&${Test/impala1host}&...
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
impala://jdbc:impala://Impala1host:21050/?...&UID=sample2&PWD=sample2&...
```

Actuellement, Cloudera Impala reconnaît les propriétés JDBC `UID` et `PWD`.

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance Cloudera Impala.

Type de gestionnaire	Classe
Gestionnaire de composites	`ImpalaCompositeHandler`
Gestionnaire de métadonnées	`ImpalaMetadataHandler`
Gestionnaire d'enregis trements	`ImpalaRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une instance Cloudera Impala unique prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`impala://jdbc:impala://Impala1host:21050/?secret=${Test/impala1host}`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant indique les types de données correspondants pour JDBC, Cloudera Impala et Arrow.

JDBC	Cloudera Impala	Flèche
Booléen	Booléen	Bit
Entier	TINYINT	Tiny
Court	SMALLINT	Smallint
Entier	INT	Int
Long	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8
Date	date	DateDay
Horodatage	timestamp	DateMilli
Chaîne	VARCHAR	Varchar
Octets	octets	Varbinary
BigDecimal	Décimal	Décimal
ARRAY	N/A (voir la remarque)	Liste

Note

Actuellement, Cloudera Impala ne prend pas en charge les types d'agrégats `ARRAY`, `MAP`, `STRUCT` ou `UNIONTYPE`. Les colonnes de types d'agrégats sont traitées comme des colonnes `VARCHAR` dans SQL.

Partitions et déversements

Les partitions sont utilisées pour déterminer comment générer des divisions pour le connecteur. Athena construit une colonne synthétique de type `varchar` qui représente le schéma de partitionnement de la table afin d'aider le connecteur à générer des divisions. Le connecteur ne modifie pas la définition réelle de la table.

Performance

Cloudera Impala prend en charge les partitions statiques. Le connecteur Athena Cloudera Impala peut récupérer les données de ces partitions en parallèle. Si vous souhaitez interroger des jeux de données très volumineux avec une distribution de partition uniforme, le partitionnement statique est fortement recommandé. Le connecteur Cloudera Impala résiste à la limitation due à la simultanéité.

Le connecteur Athena Cloudera Impala effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Les clauses `LIMIT`, les prédicats simples et les expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution des requêtes.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena Cloudera Impala peut combiner ces expressions et les pousser directement vers Cloudera Impala pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena Cloudera Impala suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%')
LIMIT 10;
```

Requêtes passthrough

Le connecteur Cloudera Impala prend en charge les requêtes directes. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Cloudera Impala, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Cloudera Impala. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les informations les plus récentes sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur Cloudera Impala sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena CloudWatch

Le CloudWatch connecteur Amazon Athena permet à Amazon Athena de communiquer CloudWatch avec vous afin que vous puissiez interroger les données de vos journaux avec SQL.

Le connecteur vous met LogGroups en correspondance sous forme de schémas et chacun d'entre eux LogStream sous forme de table. Le connecteur mappe également une `all_log_streams` vue spéciale qui contient tout ce qui LogStreams se trouve dans le LogGroup. Cette vue vous permet d'interroger tous les journaux en une seule fois LogGroup au lieu de les parcourir LogStream individuellement.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le CloudWatch connecteur.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.
- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.
- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.

- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.

Le connecteur prend également en charge le contrôle de congestion AIMD pour gérer les événements de régulation CloudWatch via la construction du SDK Amazon Athena Query Federation. `ThrottlingInvoker` Vous pouvez modifier le comportement de limitation par défaut en définissant l'une des variables d'environnement facultatives suivantes :

- throttle_initial_delay_ms – Le délai d'appel initial appliqué après le premier événement de congestion. La valeur par défaut est de 10 millisecondes.
- throttle_max_delay_ms – Le délai maximal entre les appels. Vous pouvez obtenir le TPS en le divisant en 1 000 ms. La valeur par défaut est de 1 000 millisecondes.
- throttle_dimine_factor – Le facteur par lequel Athena réduit le taux d'appels. La valeur par défaut est 0,5.
- throttle_increase_ms – La vitesse à laquelle Athena réduit le délai d'appel. La valeur par défaut est de 10 millisecondes.

Base de données et tables

Le CloudWatch connecteur Athena vous met en correspondance LogGroups sous forme de schémas (c'est-à-dire de bases de données) et chacun d'entre eux LogStream sous forme de table. Le connecteur mappe également une `all_log_streams` vue spéciale qui contient tout ce qui LogStreams se trouve dans le LogGroup. Cette vue vous permet d'interroger tous les journaux en une seule fois LogGroup au lieu de les parcourir LogStream individuellement.

Chaque table mappée par le connecteur CloudWatch Athena possède le schéma suivant. Ce schéma correspond aux champs fournis par CloudWatch Logs.

- log_stream — Un `VARCHAR` qui contient le nom de la ligne d' LogStream où provient la ligne.

- time (temps) – Un `INT64` qui contient l'heure à laquelle la ligne de journal a été générée.
- message – Un `VARCHAR` qui contient le message du journal.

Exemples

L'exemple suivant montre comment exécuter une `SELECT` requête sur un objet spécifique LogStream.

```
SELECT *
FROM "lambda:cloudwatch_connector_lambda_name"."log_group_path"."log_stream_name"
LIMIT 100
```

L'exemple suivant montre comment utiliser la `all_log_streams` vue pour exécuter une requête sur tous les éléments LogStreams d'une valeur spécifiée LogGroup.

```
SELECT *
FROM "lambda:cloudwatch_connector_lambda_name"."log_group_path"."all_log_streams"
LIMIT 100
```

Autorisations nécessaires

Pour plus de détails sur les politiques IAM requises par ce connecteur, reportez-vous à la section `Policies` du fichier athena-cloudwatch.yaml. La liste suivante résume les autorisations requises.

- Amazon S3 write access (Accès en écriture Amazon S3) – Le connecteur nécessite un accès en écriture à un emplacement dans Amazon S3 pour déverser les résultats à partir de requêtes volumineuses.
- Athena GetQueryExecution — Le connecteur utilise cette autorisation pour échouer rapidement lorsque la requête Athena en amont est terminée.
- CloudWatch Lecture/écriture des journaux : le connecteur utilise cette autorisation pour lire les données de vos journaux et écrire ses journaux de diagnostic.

Performance

Le CloudWatch connecteur Athena tente d'optimiser les requêtes CloudWatch en parallélisant les analyses des flux de log requis pour votre requête. Pour certains filtres temporels, le transfert des prédicats est effectué à la fois dans la fonction Lambda et dans Logs. CloudWatch

Pour des performances optimales, n'utilisez que des minuscules pour vos noms de groupes de journaux et de flux de journaux. L'utilisation d'une casse mixte oblige le connecteur à effectuer une recherche insensible à la casse, ce qui demande plus de temps de calcul.

Requêtes passthrough

Le CloudWatch connecteur prend en charge les requêtes passthrough qui utilisent la syntaxe de requête CloudWatch Logs Insights. Pour plus d'informations sur CloudWatch Logs Insights, consultez Analyser les données des CloudWatch journaux avec Logs Insights dans le guide de l'utilisateur Amazon CloudWatch Logs.

Pour créer des requêtes directes avec CloudWatch, utilisez la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            STARTTIME => 'start_time',
            ENDTIME => 'end_time',
            QUERYSTRING => 'query_string',
            LOGGROUPNAMES => 'log_group-names',
            LIMIT => 'max_number_of_results'
        ))
```

L'exemple de requête CloudWatch directe suivant filtre le `duration` champ lorsqu'il n'est pas égal à 1000.

```
SELECT * FROM TABLE(
        system.query(
            STARTTIME => '1710918615308',
            ENDTIME => '1710918615972',
            QUERYSTRING => 'fields @duration | filter @duration != 1000',
            LOGGROUPNAMES => '/aws/lambda/cloudwatch-test-1',
            LIMIT => '2'
            ))
```

Informations de licence

Le projet de CloudWatch connecteur Amazon Athena est concédé sous licence Apache-2.0.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena Metrics CloudWatch

Le connecteur Amazon Athena CloudWatch Metrics permet à Amazon Athena d' CloudWatch interroger les données de métriques avec SQL.

Pour plus d'informations sur la publication de métriques de requêtes CloudWatch depuis Athena elle-même, consultez. Contrôle des coûts et surveillance des requêtes à l'aide de CloudWatch métriques et d'événements

Prérequis

- Déployez le connecteur sur votre Compte AWS à l’aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d’informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur CloudWatch Metrics.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.
- spill_prefix – (Facultatif) Par défaut, il s’agit d’un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d’heures prédéterminé.
- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.
- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l’ID d’une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers

S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.

Le connecteur prend également en charge le contrôle de congestion AIMD pour gérer les événements de régulation CloudWatch via la construction du SDK Amazon Athena Query Federation. `ThrottlingInvoker` Vous pouvez modifier le comportement de limitation par défaut en définissant l'une des variables d'environnement facultatives suivantes :

- throttle_initial_delay_ms – Le délai d'appel initial appliqué après le premier événement de congestion. La valeur par défaut est de 10 millisecondes.
- throttle_max_delay_ms – Le délai maximal entre les appels. Vous pouvez obtenir le TPS en le divisant en 1 000 ms. La valeur par défaut est de 1 000 millisecondes.
- throttle_dimine_factor – Le facteur par lequel Athena réduit le taux d'appels. La valeur par défaut est 0,5.
- throttle_increase_ms – La vitesse à laquelle Athena réduit le délai d'appel. La valeur par défaut est de 10 millisecondes.

Base de données et tables

Le connecteur Athena CloudWatch Metrics mappe vos espaces de noms, dimensions, métriques et valeurs métriques dans deux tables dans un schéma unique appelé. `default`

Le tableau des métriques

La table `metrics` contient les métriques disponibles telles que définies de manière unique par une combinaison d'espace de noms, d'ensemble et de nom. La table `metrics` contient les colonnes suivantes.

- namespace (espace de noms) – Un `VARCHAR` contenant l'espace de noms.
- metric_name– Un `VARCHAR` contenant le nom de la métrique.
- dimensions – Un `LIST` des objets `STRUCT` composés de `dim_name (VARCHAR)` et de `dim_value (VARCHAR)`.
- statistic (statistique) – Un `LIST` de statistiques `VARCH` (par exemple, `p90`, `AVERAGE`, ...) disponible pour la métrique.

La table metric_samples

La table `metric_samples` contient les échantillons de métriques disponibles pour chaque métrique de la table `metrics`. La table `metric_samples` contient les colonnes suivantes.

- namespace (espace de noms) – Un `VARCHAR` qui contient l'espace de noms.
- metric_name– Un `VARCHAR` qui contient le nom de la métrique.
- dimensions – Un `LIST` des objets `STRUCT` composés de `dim_name (VARCHAR)` et de `dim_value (VARCHAR)`.
- dim_name– Un champ pratique `VARCHAR` que vous pouvez utiliser pour filtrer facilement en fonction du nom d'une dimension unique.
- dim_value– Un champ pratique `VARCHAR` que vous pouvez utiliser pour filtrer facilement en fonction du nom d'une valeur unique.
- period (période) – Un champ `INT` qui représente la « période » de la métrique en secondes (par exemple, une métrique de 60 secondes).
- timestamp (horodatage) – Un champ `BIGINT` qui représente l'époque en secondes à laquelle se rapporte l'exemple de métrique.
- value (valeur) – Un champ `FLOAT8` qui contient la valeur de l'exemple.
- statistic (statistique) – Un `VARCHAR` qui contient le type statistique de l'exemple (par exemple, `AVERAGE` ou `p90`).

Autorisations nécessaires

Pour plus de détails sur les politiques IAM requises par ce connecteur, consultez la `Policies` section du athena-cloudwatch-metricsfichier .yaml. La liste suivante résume les autorisations requises.

- Amazon S3 write access (Accès en écriture Amazon S3) – Le connecteur nécessite un accès en écriture à un emplacement dans Amazon S3 pour déverser les résultats à partir de requêtes volumineuses.
- Athena GetQueryExecution — Le connecteur utilise cette autorisation pour échouer rapidement lorsque la requête Athena en amont est terminée.
- CloudWatch Métriques ReadOnly — Le connecteur utilise cette autorisation pour interroger les données de vos métriques.

- CloudWatch Enregistrement des journaux : le connecteur utilise cet accès pour écrire ses journaux de diagnostic.

Performance

Le connecteur Athena CloudWatch Metrics tente d'optimiser les requêtes par rapport aux CloudWatch métriques en parallélisant les analyses des flux de log requis pour votre requête. Pour certains filtres de période, de métrique, d'espace de noms et de dimension, le transfert des prédicats est effectué à la fois dans la fonction Lambda et dans Logs. CloudWatch

Informations de licence

Le projet de connecteur Amazon Athena CloudWatch Metrics est concédé sous licence Apache-2.0.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur CMDB Amazon Athena AWS

Le AWS connecteur Amazon Athena CMDB permet à Athena de communiquer avec différents AWS services afin que vous puissiez les interroger avec SQL.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur AWS CMDB.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.
- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de

stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.

- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.
- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.
- default_ec2_image_owner – (Facultatif) Lorsque ce paramètre est défini, il contrôle le propriétaire de l'image Amazon EC2 par défaut qui filtre les Amazon Machine Images (AMI). Si vous ne définissez pas cette valeur et que votre requête par rapport à la table des images EC2 n'inclut pas de filtre pour le propriétaire, vos résultats incluront toutes les images publiques.

Bases de données et tables

Le connecteur Athena AWS CMDB met à disposition les bases de données et tables suivantes pour interroger votre inventaire de ressources. AWS Pour plus d'informations sur les colonnes disponibles dans chaque table, exécutez une instruction `DESCRIBE` *`database`*`.`*`table`*à l'aide de la console Athena ou de l'API.

- ec2 – Cette base de données contient des ressources relatives à Amazon EC2, notamment les suivantes.

- ebs_volumes – Contient les détails de vos volumes Amazon EBS.
- ec2_instances – Contient les détails de vos instances EC2.
- ec2_images – Contient les détails des images de votre instance EC2.
- routing_tables – Contient les détails de vos tables de routage VPC.

- security_groups – Contient les détails de vos groupes de sécurité.
- sous-réseaux – Contient les détails de vos sous-réseaux VPC.
- vpc – Contient les détails de vos VPC.

- mr – Cette base de données contient des ressources relatives à Amazon EMR, notamment les suivantes.

- emr_clusters – Contient les détails de vos clusters EMR.

- rds – Cette base de données contient des ressources relatives à Amazon RDS, notamment les suivantes.

- rds_instances – Contient les détails de vos instances RDS.

- s3 – Cette base de données contient des ressources relatives à RDS, notamment les suivantes.

- buckets (compartiment)s – Contient les détails de vos compartiments Amazon S3.
- objects (objets) – Contient les détails de vos objets Amazon S3, à l'exception de leur contenu.

Autorisations nécessaires

Pour plus de détails sur les politiques IAM requises par ce connecteur, consultez la `Policies` section du athena-aws-cmdbfichier .yaml. La liste suivante résume les autorisations requises.

- Amazon S3 write access (Accès en écriture Amazon S3) – Le connecteur nécessite un accès en écriture à un emplacement dans Amazon S3 pour déverser les résultats à partir de requêtes volumineuses.
- Athena GetQueryExecution — Le connecteur utilise cette autorisation pour échouer rapidement lorsque la requête Athena en amont est terminée.
- S3 List (Liste S3) – Le connecteur utilise cette autorisation pour répertorier vos compartiments et vos objets Amazon S3.
- EC2 Describe (Description EC2) – Le connecteur utilise cette autorisation pour décrire des ressources telles que vos instances Amazon EC2, vos groupes de sécurité, vos VPC et vos volumes Amazon EBS.

- EMR Describre/Liste (Description EMR/Liste) – Le connecteur utilise cette autorisation pour décrire vos clusters EMR.
- RDS Describe (Description RDS) – Le connecteur utilise cette autorisation pour décrire vos instances RDS.

Performance

Actuellement, le connecteur Athena AWS CMDB ne prend pas en charge les scans parallèles. La poussée vers le bas de prédicat est effectuée au sein de la fonction Lambda. Dans la mesure du possible, des prédicats partiels sont transférés aux services interrogés. Par exemple, une requête pour obtenir les détails d'une instance Amazon EC2 spécifique appelle l'API EC2 avec l'ID d'instance spécifique pour exécuter une opération de description ciblée.

Informations de licence

Le projet de connecteur Amazon Athena AWS CMDB est concédé sous licence Apache-2.0.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Db2 IBM

Le connecteur Amazon Athena pour Db2 permet à Amazon Athena d'exécuter des requêtes SQL sur vos bases de données Db2 IBM à l'aide de JDBC.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.
- Avant d'utiliser ce connecteur, configurez un VPC et un groupe de sécurité. Pour plus d'informations, consultez Création d'un VPC pour un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.

- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Les types de données de date et d'horodatage dans des conditions de filtre doivent être convertis en types de données appropriés.

Conditions

Les termes suivants concernent le connecteur Db2.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette rubrique pour configurer le connecteur Db2.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
dbtwo://${jdbc_connection_string}
```

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`Db2MuxCompositeHandler`
Gestionnaire de métadonnées	`Db2MuxMetadataHandler`
Gestionnaire d'enregistrements	`Db2MuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
`$`*`catalog`*`_connection_string`	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `mydbtwocatalog` , le nom de la variable d'environnement est alors `mydbtwocatalog_connection_string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *`AWS_LAMBDA_FUNCTION_NAME`* `}`.

Les exemples de propriétés suivants concernent une fonction Lambda Db2 MUX qui prend en charge deux instances de base de données : `dbtwo1` (par défaut) et `dbtwo2`.

Propriété	Valeur
`default`	`dbtwo://jdbc:db2://dbtwo1.hostname:port/ database_name :${secret1_name }`
`dbtwo_catalog1_connection_string`	`dbtwo://jdbc:db2://dbtwo1. hostname:port/ database_name :${secret1_name }`
`dbtwo_catalog2_connection_string`	`dbtwo://jdbc:db2://dbtwo2. hostname:port/ database_name :${secret2_name }`

Fourniture des informations d'identification

Pour fournir un nom d'utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${secret_name}`.

```
dbtwo://jdbc:db2://hostname:port/database_name:${secret_name}
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
dbtwo://jdbc:db2://hostname:port/database_name:user=user_name;password=password;
```

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrement suivants pour vous connecter à une seule instance Db2.

Type de gestionnaire	Classe
Gestionnaire de composites	`Db2CompositeHandler`
Gestionnaire de métadonnées	`Db2MetadataHandler`
Gestionnaire d'enregistrements	`Db2RecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une seule instance Db2 prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	dbtwo://jdbc:db2://hostname:port/*`database_name`* :${secret_name}

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant affiche les types de données correspondants pour JDBC et Arrow.

Db2	Flèche
CHAR	VARCHAR
VARCHAR	VARCHAR
DATE	DATEDAY
TIME	VARCHAR
TIMESTAMP	DATEMILLI
DATETIME	DATEMILLI
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	DECIMAL
REAL	FLOAT8
DOUBLE	FLOAT8
DECFLOAT	FLOAT8

Partitions et déversements

Une partition est représentée par une ou plusieurs colonnes de partition de type `varchar`. Le connecteur Db2 crée des partitions en utilisant les schémas d'organisation suivants.

- Distribution par hachage
- Partition par gamme

- Organisation par dimensions

Le connecteur récupère les détails des partitions, tels que le nombre de partitions et le nom des colonnes, à partir d'une ou plusieurs tables de métadonnées Db2. Les divisions sont créées en fonction du nombre de partitions identifiées.

Performance

Le connecteur Athena Db2 effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Les clauses `LIMIT`, les prédicats simples et les expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution des requêtes.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena Db2 peut combiner ces expressions et les pousser directement vers Db2 pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena Db2 suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%')
LIMIT 10;
```

Requêtes directes

Le connecteur DB2 prend en charge les requêtes passthrough. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Db2, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Db2. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les informations les plus récentes sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur DB2 sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena IBM Db2 AS/400 (Db2 iSeries)

Le connecteur Amazon Athena pour Db2 AS/400 permet à Amazon Athena d'exécuter des requêtes SQL sur vos bases de données IBM Db2 AS/400 (Db2 iSeries) à l'aide de JDBC.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.
- Avant d'utiliser ce connecteur, configurez un VPC et un groupe de sécurité. Pour plus d'informations, consultez Création d'un VPC pour un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Les types de données de date et d'horodatage dans des conditions de filtre doivent être convertis en types de données appropriés.

Conditions

Les termes suivants concernent le connecteur Db2 AS/400.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.

- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Db2 AS/400.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
db2as400://${jdbc_connection_string}
```

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`Db2MuxCompositeHandler`
Gestionnaire de métadonnées	`Db2MuxMetadataHandler`

Handler (Gestionnaire)	Classe
Gestionnaire d'enregistrements	`Db2MuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
`$`*`catalog`*`_connection_string`	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `mydb2as400catalog` , le nom de la variable d'environnement est alors `mydb2as400catalog_connection_string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *`AWS_LAMBDA_FUNCTION_NAME`* `}`.

Les exemples de propriétés suivants concernent une fonction Lambda Db2 MUX qui prend en charge deux instances de base de données : `db2as4001` (par défaut) et `db2as4002`.

Propriété	Valeur
`default`	`db2as400://jdbc:as400://` *`<ip_address>`* `;`*`<properties>`* `;:${`*`<secret name>`*`};`
`db2as400_catalog1_connection_string`	`db2as400://jdbc:as400://db2as4001.hostname/:${` *`secret1_name`* `}`
`db2as400_catalog2_connection_string`	`db2as400://jdbc:as400://db2as4002.hostname/:${` *`secret2_name`* `}`

Propriété	Valeur
`db2as400_catalog3_connection_string`	`db2as400://jdbc:as400://` *`<ip_address>`* `;user=`*`<username>`* `;password=` *`<password>`* `;`*`<properties>`* `;`

Fourniture des informations d’identification

Pour fournir un nom d’utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

  Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n’utilise pas Amazon RDS, stockez les informations d’identification au format JSON au format suivant :

  ```
  {"username": "${username}", "password": "${password}"}
  ```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${secret_name}`.

```
db2as400://jdbc:as400://<ip_address>;<properties>;:${<secret_name>};
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
db2as400://jdbc:as400://<ip_address>;user=<username>;password=<password>;<properties>;
```

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance DB2 AS/400.

Type de gestionnaire	Classe
Gestionnaire de composites	`Db2CompositeHandler`
Gestionnaire de métadonnées	`Db2MetadataHandler`
Gestionnaire d'enregis trements	`Db2RecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une seule instance Db2 AS/400 prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`db2as400://jdbc:as400://` *`<ip_address>`* `;`*`<properties>`* `;:` `${`*`<secret_name>`* `};`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant montre les types de données correspondants pour JDBC et Apache Arrow.

DB2 AS/400	Flèche
CHAR	VARCHAR
VARCHAR	VARCHAR

DB2 AS/400	Flèche
DATE	DATEDAY
TIME	VARCHAR
TIMESTAMF	DATEMILLI
DATETIME	DATEMILLI
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	DECIMAL
REAL	FLOAT8
DOUBLE	FLOAT8
DECFLOAT	FLOAT8

Partitions et déversements

Une partition est représentée par une ou plusieurs colonnes de partition de type `varchar`. Le connecteur Db2 AS/400 crée des partitions selon les schémas d'organisation suivants.

- Distribution par hachage
- Partition par gamme
- Organisation par dimensions

Le connecteur récupère les détails des partitions, tels que le nombre de partitions et le nom de colonne, à partir d'une ou de plusieurs tables de métadonnées DB2 AS/400. Les divisions sont créées en fonction du nombre de partitions identifiées.

Performance

Pour améliorer les performances, utilisez le prédicat pushdown pour interroger Athena, comme dans les exemples suivants.

```
SELECT * FROM "lambda:<LAMBDA_NAME>"."<SCHEMA_NAME>"."<TABLE_NAME>"
 WHERE integercol = 2147483647
```

```
SELECT * FROM "lambda: <LAMBDA_NAME>"."<SCHEMA_NAME>"."<TABLE_NAME>"
 WHERE timestampcol >= TIMESTAMP '2018-03-25 07:30:58.878'
```

Requêtes passthrough

Le connecteur Db2 AS/400 prend en charge les requêtes passthrough. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Db2 AS/400, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans DB2 AS/400. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les informations les plus récentes sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur DB2 AS/400 sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour DocumentDB

Le connecteur Amazon Athena DocumentDB permet à Amazon Athena de communiquer avec vos instances DocumentDB afin que vous puissiez interroger vos données DocumentDB avec SQL. Le connecteur fonctionne également avec n'importe quel point de terminaison compatible avec MongoDB.

Contrairement aux magasins de données relatives traditionnels, les collections Amazon DocumentDB n'ont pas de schéma défini. DocumentDB n'a pas de magasin de métadonnées. Chaque entrée d'une collection DocumentDB peut comporter des champs et des types de données différents.

Le connecteur DocumentDB prend en charge deux mécanismes pour générer des informations de schéma de table : l'inférence de schéma de base et les métadonnées. AWS Glue Data Catalog

L'inférence de schéma est la valeur par défaut. Cette option analyse un petit nombre de documents de votre collection, réunit tous les champs et impose des champs dont les types de données ne se chevauchent pas. Cette option fonctionne bien pour les collections dont les entrées sont généralement uniformes.

Pour les collections comportant une plus grande variété de types de données, le connecteur prend en charge la récupération de métadonnées à partir du AWS Glue Data Catalog. Si le connecteur détecte une AWS Glue base de données et une table qui correspondent aux noms de votre base de données et de collection DocumentDB, il obtient ses informations de schéma dans la table correspondante AWS Glue . Lorsque vous créez votre AWS Glue table, nous vous recommandons d'en faire un sur-ensemble de tous les champs auxquels vous souhaitez accéder depuis votre collection DocumentDB.

Si Lake Formation est activé sur votre compte, le rôle IAM de votre connecteur Lambda fédéré Athena que vous avez déployé dans AWS Serverless Application Repository le doit avoir un accès en lecture dans Lake Formation au. AWS Glue Data Catalog

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un

connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur DocumentDB.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.
- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.
- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.
- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.
- disable_glue — (Facultatif) S'il est présent et défini sur true, le connecteur ne tente pas de récupérer des métadonnées supplémentaires à partir de. AWS Glue
- glue_catalog – (Facultatif) Utilisez cette option pour spécifier un catalogue AWS Glue entre compte. Par défaut, le connecteur tente d'obtenir des métadonnées à partir de son propre AWS Glue compte.
- default_docdb – S'il est présent, il spécifie une chaîne de connexion DocumentDB à utiliser lorsqu'aucune variable d'environnement spécifique au catalogue n'existe.

- disable_projection_and_casing – (Facultatif) Désactive la projection et la casse. À utiliser si vous souhaitez interroger des tables Amazon DocumentDB qui utilisent des noms de colonnes sensibles à la casse. Le paramètre `disable_projection_and_casing` utilise les valeurs suivantes pour spécifier le comportement de la casse et du mappage des colonnes :
  - false (faux) : il s'agit du paramètre par défaut. La projection est activée et le connecteur s'attend à ce que tous les noms de colonnes soient en minuscules.
  - true (vrai) : désactive la projection et la casse. Lorsque vous utilisez le paramètre `disable_projection_and_casing`, gardez à l'esprit les points suivants :
    - L'utilisation de ce paramètre peut augmenter l'utilisation de la bande passante. En outre, si votre fonction Lambda ne se trouve pas dans la même Région AWS que votre source de données, vous devrez supporter des coûts de transfert interrégionaux AWS standard plus élevés en raison de l'utilisation plus importante de la bande passante. Pour plus d'informations sur les coûts de transfert entre régions, consultez la section Frais de transfert de AWS données pour les architectures serveur et sans serveur sur le blog du réseau AWS partenaire.
    - Étant donné qu'un plus grand nombre d'octets est transféré et qu'un plus grand nombre d'octets nécessite un délai de désérialisation plus long, la latence globale peut augmenter.
- enable_case_insensitive_match — (Facultatif) Lorsque`true`, effectue des recherches sans distinction majuscules/minuscules sur les noms de schéma et de table dans Amazon DocumentDB. L'argument par défaut est `false`. À utiliser si votre requête contient des noms de schéma ou de table en majuscules.

Définition des chaînes de connexion

Vous pouvez fournir une ou plusieurs propriétés qui définissent les détails de connexion DocumentDB pour les instances de DocumentDB que vous utilisez avec le connecteur. Pour ce faire, définissez une variable d'environnement Lambda qui correspond au nom du catalogue que vous souhaitez utiliser dans Athena. Supposons, par exemple, que vous souhaitiez utiliser les requêtes suivantes pour interroger deux instances DocumentDB différentes depuis Athena :

```
SELECT * FROM "docdb_instance_1".database.table
```

```
SELECT * FROM "docdb_instance_2".database.table
```

Avant de pouvoir utiliser ces deux instructions SQL, vous devez ajouter deux variables d'environnement à votre fonction Lambda : `docdb_instance_1` et `docdb_instance_2`. La valeur pour chacune d'elles doit être une chaîne de connexion DocumentDB au format suivant :

```
mongodb://:@:/?ssl=true&ssl_ca_certs=rds-combined-ca-bundle.pem&replicaSet=rs0
```

Utilisation de secrets

Vous pouvez éventuellement utiliser AWS Secrets Manager une partie ou la totalité de la valeur pour les détails de votre chaîne de connexion. Pour utiliser la fonctionnalité de requête fédérée d'Athena avec Secrets Manager, le VPC connecté à votre fonction Lambda doit avoir un accès internet ou un point de terminaison de VPC pour vous connecter à Secrets Manager.

Si vous utilisez la syntaxe `${my_secret}` pour indiquer le nom d'un secret provenant de Secrets Manager dans votre chaîne de connexion, le connecteur remplace exactement `${my_secret}` par sa valeur en texte brut provenant de Secrets Manager. Les secrets doivent être stockés sous forme de texte brut avec la valeur *<username>*`:`*<password>*. Les secrets stockés sous la forme `{username:`*<username>*`,password:`*<password>*`}` ne seront pas transmis correctement à la chaîne de connexion.

Les secrets peuvent également être utilisés pour l'ensemble de la chaîne de connexion, le nom d'utilisateur et le mot de passe pouvant être définis dans le secret.

Supposons, par exemple, que vous définissiez la variable d'environnement Lambda pour `docdb_instance_1` sur la valeur suivante :

```
mongodb://${docdb_instance_1_creds}@myhostname.com:123/?ssl=true&ssl_ca_certs=rds-
combined-ca-bundle.pem&replicaSet=rs0
```

Le SDK Athena Query Federation tente automatiquement de récupérer un secret nommé `docdb_instance_1_creds` à partir de Secrets Manager et d'injecte cette valeur à la place de `${docdb_instance_1_creds}`. Toute partie de la chaîne de connexion qui est entourée par la combinaison de caractères `${ }` est interprétée comme un secret de Secrets Manager. Si vous spécifiez un nom de secret que le connecteur ne trouve pas dans Secrets Manager, le connecteur ne remplace pas le texte.

Configuration de bases de données et de tables dans AWS Glue

Étant donné que la fonctionnalité d'inférence de schéma intégrée du connecteur scanne un nombre limité de documents et ne prend en charge qu'un sous-ensemble de types de données, vous pouvez préférer l'utiliser AWS Glue pour les métadonnées.

Pour activer une AWS Glue table à utiliser avec Amazon DocumentDB, vous devez disposer d'une base de AWS Glue données et d'une table pour la base de données et la collection DocumentDB pour lesquelles vous souhaitez fournir des métadonnées supplémentaires.

Pour utiliser une AWS Glue table pour des métadonnées supplémentaires

1. Lorsque vous modifiez la table et la base de données dans la AWS Glue console, ajoutez la propriété de table suivante.

   - docdb-metadata-flag— Cette propriété indique au connecteur DocumentDB que le connecteur peut utiliser la table pour des métadonnées supplémentaires. Vous pouvez fournir n'importe quelle valeur pour `docdb-metadata-flag` tant que la propriété `docdb-metadata-flag` est présente dans la liste des propriétés de la table.

2. (Facultatif) Ajoutez la propriété de table sourceTable. Cette propriété définit le nom de la table source dans Amazon DocumentDB. Utilisez cette propriété si les règles de dénomination des AWS Glue tables vous empêchent de créer une AWS Glue table portant le même nom que votre table Amazon DocumentDB. Par exemple, les majuscules ne sont pas autorisées dans les noms de tables AWS Glue , mais elles le sont dans les noms de tables Amazon DocumentDB.

3. (Facultatif) Ajoutez la propriété de table columnMapping. Cette propriété définit les mappages de noms de colonnes. Utilisez cette propriété si les règles de dénomination des AWS Glue colonnes vous empêchent de créer une AWS Glue table portant les mêmes noms de colonne que ceux de votre table Amazon DocumentDB. Cela peut être utile, car les majuscules sont autorisées dans les noms de colonnes Amazon DocumentDB, mais elles ne le sont pas dans les noms de colonnes AWS Glue .

   La valeur de la propriété `columnMapping` est censée être un ensemble de mappages au format `col1=Col1,col2=Col2`.

   > **Note**
   >
   > Le mappage des colonnes s'applique uniquement aux noms de colonnes de niveau supérieur et non aux champs imbriqués.

   Après avoir ajouté la propriété AWS Glue `columnMapping` table, vous pouvez supprimer la variable d'environnement `disable_projection_and_casing` Lambda.

4. Assurez-vous d'utiliser les types de données appropriés AWS Glue tels que listés dans ce document.

Prise en charge du type de données

Cette section répertorie les types de données que le connecteur DocumentDB utilise pour l'inférence de schéma, ainsi que les types de données lorsque des AWS Glue métadonnées sont utilisées.

Types de données d’inférence de schéma

La fonction d’inférence de schéma du connecteur DocumentDB tente de déduire des valeurs comme appartenant à l’un des types de données suivants. Le tableau indique les types de données correspondants pour Amazon DocumentDB, Java et Apache Arrow.

Apache	Java ou DocDB
VARCHAR	Chaîne
INT	Entier
BIGINT	Long
BIT	Booléen
FLOAT4	Float
FLOAT8	Double
TIMESTAMPSEC	Date
VARCHAR	ObjectId
LIST	Liste
STRUCT	Document

AWS Glue types de données

Si vous utilisez AWS Glue des métadonnées supplémentaires, vous pouvez configurer les types de données suivants. Le tableau indique les types de données correspondants pour AWS Glue et Apache Arrow.

AWS Glue	Apache
int	INT
bigint	BIGINT
double	FLOAT8
float	FLOAT4
boolean	BIT
binary	VARBINARY
chaîne	VARCHAR
Liste	LIST
Struct	STRUCT

Autorisations nécessaires

Pour plus de détails sur les politiques IAM requises par ce connecteur, reportez-vous à la section `Policies` du fichier athena-docdb.yaml. La liste suivante résume les autorisations requises.

- Amazon S3 write access (Accès en écriture Amazon S3) – Le connecteur nécessite un accès en écriture à un emplacement dans Amazon S3 pour déverser les résultats à partir de requêtes volumineuses.
- Athena GetQueryExecution — Le connecteur utilise cette autorisation pour échouer rapidement lorsque la requête Athena en amont est terminée.
- AWS Glue Data Catalog— Le connecteur DocumentDB nécessite un accès en lecture seule au pour AWS Glue Data Catalog obtenir des informations sur le schéma.

- CloudWatch Journaux : le connecteur a besoin d'accéder aux CloudWatch journaux pour stocker les journaux.
- AWS Secrets Manager accès en lecture — Si vous choisissez de stocker les détails du point de terminaison DocumentDB dans Secrets Manager, vous devez autoriser le connecteur à accéder à ces secrets.
- Accès VPC – Le connecteur nécessite la possibilité d'attacher des interfaces à votre VPC et de les détacher afin qu'il puisse s'y connecter et communiquer avec vos instances DocumentDB.

Performance

Le connecteur Amazon DocumentDB d'Athena ne prend pas actuellement en charge les analyses parallèles, mais tente de pousser les prédicats vers le bas dans le cadre de ses requêtes DocumentDB, et les prédicats contre les index de votre collection DocumentDB entraînent une réduction significative des données analysées.

La fonction Lambda effectue une poussée vers le bas de projection pour réduire les données analysées par la requête. Cependant, la sélection d'un sous-ensemble de colonnes entraîne parfois un temps d'exécution plus long de la requête. Les clauses `LIMIT` réduisent la quantité de données analysées, mais si vous ne fournissez pas de prédicat, vous devez vous attendre à ce que les requêtes `SELECT` avec une clause `LIMIT` analysent au moins 16 Mo de données.

Requêtes passthrough

Le connecteur Athena Amazon DocumentDB prend en charge les requêtes passthrough et est basé sur NoSQL. Pour plus d'informations sur l'interrogation d'Amazon DocumentDB, consultez la section Requête dans le manuel du développeur Amazon DocumentDB.

Pour utiliser des requêtes directes avec Amazon DocumentDB, utilisez la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            database => 'database_name',
            collection => 'collection_name',
            filter => '{query_syntax}'
        ))
```

L'exemple suivant interroge la `example` base de données de la `TPCDS` collection, en filtrant tous les livres portant le titre Bill of Rights.

```
SELECT * FROM TABLE(
        system.query(
            database => 'example',
            collection => 'tpcds',
            filter => '{title: "Bill of Rights"}'
        ))
```

Ressources supplémentaires

- Pour un article sur l'utilisation d'Amazon Athena Federated Query pour connecter une base de données MongoDB à QuickSightAmazon afin de créer des tableaux de bord et des visualisations, consultez Visualiser les données MongoDB d'Amazon à l' QuickSight aide d'Amazon Athena Federated Query sur le blog Big Data.AWS
- Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour DynamoDB

Le connecteur Amazon Athena DynamoDB permet à Amazon Athena de communiquer avec DynamoDB afin que vous puissiez requêter vos tables avec SQL. Les opérations d'écriture telles que INSERT INTO ne sont pas prises en charge.

Si Lake Formation est activé sur votre compte, le rôle IAM de votre connecteur Lambda fédéré Athena que vous avez déployé dans AWS Serverless Application Repository le doit avoir un accès en lecture dans Lake Formation au. AWS Glue Data Catalog

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur DynamoDB.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.

- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.
- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.
- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.
- disable_glue — (Facultatif) S'il est présent et défini sur true, le connecteur ne tente pas de récupérer des métadonnées supplémentaires à partir de. AWS Glue
- glue_catalog – (Facultatif) Utilisez cette option pour spécifier un catalogue AWS Glue entre compte. Par défaut, le connecteur tente d'obtenir des métadonnées à partir de son propre AWS Glue compte.
- disable_projection_and_casing – (Facultatif) Désactive la projection et la casse. À utiliser si vous souhaitez interroger des tables DynamoDB dont le nom de colonne contient des encadrés et que vous ne souhaitez pas spécifier `columnMapping` de propriété sur votre table. AWS Glue

  Le paramètre `disable_projection_and_casing` utilise les valeurs suivantes pour spécifier le comportement de la casse et du mappage des colonnes :

  - auto – Désactive la projection et la casse lorsqu'un type précédemment non pris en charge est détecté et que le mappage des noms de colonnes n'est pas défini sur la table. Il s'agit du paramètre par défaut.
  - always (toujours) – Désactive la projection et la casse de manière inconditionnelle. Cela est utile lorsque vous avez des noms de vos colonnes DynamoDB sensibles à la casse, mais que vous ne souhaitez pas spécifier de mappage de noms de colonnes.

Lorsque vous utilisez le paramètre `disable_projection_and_casing`, gardez à l'esprit les points suivants :

- L'utilisation de ce paramètre peut augmenter l'utilisation de la bande passante. En outre, si votre fonction Lambda ne se trouve pas dans la même Région AWS que votre source de données, vous devrez supporter des coûts de transfert interrégionaux AWS standard plus élevés en raison de l'utilisation plus importante de la bande passante. Pour plus d'informations sur les coûts de transfert entre régions, consultez la section Frais de transfert de AWS données pour les architectures serveur et sans serveur sur le blog du réseau AWS partenaire.
- Étant donné qu'un plus grand nombre d'octets est transféré et qu'un plus grand nombre d'octets nécessite un délai de désérialisation plus long, la latence globale peut augmenter.

Configuration de bases de données et de tables dans AWS Glue

La capacité d'inférence de schéma intégrée du connecteur étant limitée, vous souhaiterez peut-être l'utiliser AWS Glue pour les métadonnées. Pour ce faire, vous devez disposer d'une base de données et d'une table AWS Glue. Pour les utiliser avec DynamoDB, vous devez modifier leurs propriétés.

Pour modifier les propriétés de base de données dans la AWS Glue console

1. Connectez-vous à la AWS Glue console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/glue/.
2. Dans le volet de navigation, développez le catalogue de données, puis choisissez Databases.

   Sur la page Databases (Bases de données), vous pouvez modifier une base de données existante ou sélectionner Add database (Ajouter une base de données) pour en créer une.
3. Dans la liste des bases de données, sélectionnez le lien de la base de données à modifier.
4. Choisissez Modifier.
5. Sur la page Mettre à jour une base de données, sous Paramètres de base de données, dans Emplacement, ajoutez la chaîne**`dynamo-db-flag`**. Ce mot clé indique que la base de données contient des tables que le connecteur Athena DynamoDB utilise pour des métadonnées supplémentaires et qu'elle est requise pour les bases de données autres que. AWS Glue `default` La propriété `dynamo-db-flag` permet de filtrer les bases de données des comptes comprenant de nombreuses bases de données.
6. Choisissez Update Database (Mettre à jour la base de données).

Pour modifier les propriétés du tableau dans la AWS Glue console

1. Connectez-vous à la AWS Glue console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/glue/.
2. Dans le volet de navigation, développez le catalogue de données, puis choisissez Tables.
3. Sur la page Tables, dans la liste des tables, choisissez le nom lié de la table que vous souhaitez modifier.
4. Sélectionnez Actions, puis Edit table (Modifier la table).
5. Sur la page Edit table (Modifier la table), dans la section Table properties (Propriétés de la table), ajoutez les propriétés de table suivantes, selon les besoins. Si vous utilisez le robot AWS Glue DynamoDB, ces propriétés sont définies automatiquement.

   - dynamoDB— Chaîne indiquant au connecteur Athena DynamoDB que la table peut être utilisée pour des métadonnées supplémentaires. Saisissez la chaîne `dynamodb` dans les propriétés de la table, dans un champ nommé classification (correspondance exacte).

     > **Note**
     >
     > La page Définir les propriétés de la table qui fait partie du processus de création de table dans la AWS Glue console comporte une section Format de données avec un champ Classification. Vous ne pouvez pas saisir ou choisir `dynamodb` ici. Après avoir créé votre table, suivez plutôt les étapes pour modifier la table et pour saisir `classification` et `dynamodb` sous forme de paire clé-valeur dans la section Propriétés de la table.

   - sourceTable – Propriété de table facultative qui définit le nom de la table source dans DynamoDB. Utilisez cette option si les règles de dénomination des AWS Glue tables vous empêchent de créer une AWS Glue table portant le même nom que votre table DynamoDB. Par exemple, les majuscules ne sont pas autorisées dans les noms de AWS Glue tables, mais elles le sont dans les noms de tables DynamoDB.
   - columnMapping – Propriété de table facultative qui définit les mappages de noms de colonnes. Utilisez cette option si les règles de dénomination des AWS Glue colonnes vous empêchent de créer une AWS Glue table portant les mêmes noms de colonne que votre table DynamoDB. Par exemple, les majuscules ne sont pas autorisées dans les noms de AWS Glue colonnes, mais le sont dans les noms de colonne DynamoDB. La valeur de la propriété devrait être au format col1=Col1,col2=Col2. Notez que le mappage des colonnes s'applique uniquement aux noms de colonnes de niveau supérieur et non aux champs imbriqués.

- defaultTimeZone— Propriété de table facultative appliquée à `date` ou à `datetime` des valeurs n'ayant pas de fuseau horaire explicite. La définition de cette valeur est recommandée pour éviter tout écart entre le fuseau horaire par défaut de la source de données et le fuseau horaire de la session Athena.
- datetimeFormatMapping— Propriété de table facultative qui spécifie le `datetime` format `date` ou à utiliser lors de l'analyse des données d'une colonne de type AWS Glue `date` or`timestamp`. Si cette propriété n'est pas spécifiée, le connecteur tente de déduire un format ISO-8601. Si le connecteur ne peut pas déduire le format `date` ou `datetime` ni analyser la chaîne brute, la valeur est alors omise du résultat.

  La valeur `datetimeFormatMapping` doit être au format `col1=someformat1,col2=someformat2`. Voici quelques exemples de formats :

  ```
  yyyyMMdd'T'HHmmss
  ddMMyyyy'T'HH:mm:ss
  ```

  Si votre colonne contient des valeurs `date` ou `datetime` sans fuseau horaire et que vous souhaitez utiliser la colonne dans la clause `WHERE`, définissez la propriété `datetimeFormatMapping` pour la colonne.

6. Si vous définissez manuellement vos colonnes, assurez-vous que vous utilisez les types de données appropriés. Si vous avez utilisé un crawler, validez les colonnes et les types qu'il a découverts.
7. Choisissez Enregistrer.

Autorisations nécessaires

Pour plus de détails sur les politiques IAM requises par ce connecteur, reportez-vous à la section `Policies` du fichier athena-dynamodb.yaml. La liste suivante résume les autorisations requises.

- Amazon S3 write access (Accès en écriture Amazon S3) – Le connecteur nécessite un accès en écriture à un emplacement dans Amazon S3 pour déverser les résultats à partir de requêtes volumineuses.
- Athena GetQueryExecution — Le connecteur utilise cette autorisation pour échouer rapidement lorsque la requête Athena en amont est terminée.
- AWS Glue Data Catalog— Le connecteur DynamoDB nécessite un accès en lecture seule pour obtenir des informations sur AWS Glue Data Catalog le schéma.

- CloudWatch Journaux : le connecteur a besoin d'accéder aux CloudWatch journaux pour stocker les journaux.
- Accès en lecture DynamoDB – Le connecteur utilise les opérations d'API `DescribeTable`, `ListSchemas`, `ListTables`, `Query` et `Scan`.

Performance

Le connecteur Athena DynamoDB prend en charge les examens parallèles et les tentatives de transfert de prédicats dans le cadre de ses requêtes DynamoDB. Un prédicat de clé de hachage avec des valeurs distinctes `X` se traduit par des appels de requête `X` à DynamoDB. Tous les autres scénarios de prédicat se traduisent par un nombre `Y` d'appels d'analyse, où `Y` est déterminé de manière heuristique en fonction de la taille de votre table et de son débit alloué. Cependant, la sélection d'un sous-ensemble de colonnes entraîne parfois un délai d'exécution plus long des requêtes.

Les clauses `LIMIT` et les prédicats simples sont poussés vers le bas, ce qui peut réduire la quantité de données analysées et entraîner une diminution du délai d'exécution des requêtes.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena DynamoDB peut combiner ces expressions et les pousser directement vers DynamoDB pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena DynamoDB suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10 and col_b < 10
LIMIT 10
```

Pour un article sur l'utilisation de la poussée vers le bas de prédicats pour améliorer les performances des requêtes fédérées, y compris DynamoDB, consultez Améliorer les requêtes fédérées avec la poussée vers le bas de prédicats dans Amazon Athena sur le blog AWS Big Data.

Requêtes directes

Le connecteur DynamoDB prend en charge les requêtes passthrough et utilise la syntaxe PartiQL. L'opération d'API GetItemDynamoDB n'est pas prise en charge. Pour plus d'informations sur l'interrogation de DynamoDB à l'aide de PartiQL, consultez les instructions PartiQL select pour DynamoDB dans le manuel Amazon DynamoDB Developer Guide.

Pour utiliser des requêtes directes avec DynamoDB, utilisez la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query_string'
        ))
```

L'exemple de requête DynamoDB passthrough suivant utilise PartiQL pour renvoyer la liste des appareils Fire TV Stick dont la propriété est postérieure au 24/12/22. `DateWatched`

```
SELECT * FROM TABLE(
        system.query(
           query => 'SELECT Devices
                       FROM WatchList
                       WHERE Devices.FireStick.DateWatched[0] > '12/24/22''
        ))
```

Résolution des problèmes

Plusieurs filtres sur une colonne de clé de tri

Message d'erreur : ne KeyConditionExpressions doit contenir qu'une seule condition par clé

Cause : ce problème peut se produire dans la version 3 du moteur Athena dans les requêtes comportant à la fois un filtre de limite inférieure et supérieure sur une colonne de clé de tri DynamoDB. DynamoDB ne prenant en charge qu'une seule condition de filtre sur une clé de tri, une erreur est générée lorsque le connecteur tente de pousser vers le bas une requête sur laquelle les deux conditions sont appliquées.

Solution : mettre à jour le connecteur vers la version 2023.11.1 ou ultérieure. Pour obtenir des instructions sur la mise à jour d'un connecteur, consultez Mise à jour d'un connecteur de source de données.

Coûts

Les coûts d'utilisation du connecteur dépendent des AWS ressources sous-jacentes utilisées. Étant donné que les requêtes qui utilisent des examens peuvent consommer un grand nombre d'unités de capacité de lecture (RCU), examinez attentivement les informations relative à la tarification Amazon DynamoDB.

Ressources supplémentaires

- Pour une introduction à l'utilisation du connecteur Amazon Athena DynamoDB, consultez Accéder aux tables Amazon DynamoDB, les interroger et les joindre à l'aide d'Athena dans le guide Modèles de recommandations AWS .
- Pour un article expliquant comment utiliser le connecteur Athena DynamoDB pour interroger des données dans DynamoDB avec SQL et visualiser des informations sur Amazon, consultez le billet de blog sur le Big AWS Data Visualize Amazon DynamoDB insights in QuickSight Amazon à l'aide du connecteur Amazon Athena DynamoDB et. QuickSight AWS Glue
- Pour un article sur l'utilisation du connecteur Amazon Athena DynamoDB avec Amazon DynamoDB, Athena et Amazon pour créer un tableau de bord de gouvernance simple, consultez le billet du blog Big AWS Data Query cross-account QuickSight Amazon DynamoDB tables using Amazon Athena Federated Query.
- Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Google BigQuery

Le connecteur Amazon Athena pour Google BigQuerypermet à Amazon Athena d'exécuter des requêtes SQL sur vos données Google. BigQuery

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les fonctions Lambda ont un délai d'expiration maximal de 15 minutes. Chaque division exécute une requête BigQuery et doit se terminer avec suffisamment de temps pour stocker les résultats afin qu'Athéna puisse les lire. Si le délai imparti à la fonction Lambda expire, la requête échoue.
- Google BigQuery fait la distinction majuscules/minuscules. Le connecteur tente de corriger la casse des noms de jeux de données et des noms de tables, mais n'effectue aucune correction de casse pour les ID de projet. Cette opération est nécessaire, car Athena met en minuscules toutes les métadonnées. Ces corrections font passer de nombreux appels supplémentaires à Google BigQuery.
- Les types de données binaires ne sont pas pris en charge.
- En raison de la BigQuery simultanéité de Google et des limites de quotas, le connecteur peut rencontrer des problèmes liés aux limites de quotas Google. Pour éviter ces problèmes, soumettez autant de contraintes BigQuery que possible à Google. Pour plus d'informations sur BigQuery les quotas, consultez la section Quotas et limites dans la BigQuery documentation de Google.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Google BigQuery .

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.
- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de

stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.

- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.
- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.
- gcp_project_id – L'ID du projet (et non le nom du projet) qui contient les jeux de données à partir desquels le connecteur doit lire (par exemple, `semiotic-primer-1234567`).
- secret_manager_gcp_creds_name — Le nom du secret AWS Secrets Manager qui contient vos informations d'identification au format JSON (par exemple,). BigQuery `GoogleCloudPlatformCredentials`
- big_query_endpoint — (Facultatif) L'URL d'un point de terminaison privé. BigQuery Utilisez ce paramètre lorsque vous souhaitez accéder BigQuery via un point de terminaison privé.

Divisions et vues

Étant donné que le BigQuery connecteur utilise l'API BigQuery Storage Read pour interroger les tables et que l'API BigQuery Storage ne prend pas en charge les vues, le connecteur utilise le BigQuery client avec une seule division pour les vues.

Performance

Pour interroger les tables, le BigQuery connecteur utilise l'API BigQuery Storage Read, qui utilise un protocole basé sur le RPC qui fournit un accès rapide au stockage BigQuery géré. Pour plus d'informations sur l'API BigQuery Storage Read, consultez la section Utiliser l'API BigQuery Storage Read pour lire les données des tables dans la documentation de Google Cloud.

La sélection d'un sous-ensemble de colonnes accélère considérablement l'exécution des requêtes et réduit le nombre de données analysées. Le connecteur est sujet à des échecs de requête lorsque la simultanéité augmente, et est généralement un connecteur lent.

Le BigQuery connecteur Google Athena effectue le transfert des prédicats vers le bas afin de réduire le nombre de données scannées par la requête. `LIMIT`les `ORDER BY` clauses, les prédicats simples et les expressions complexes sont transférés vers le connecteur afin de réduire la quantité de données numérisées et le temps d'exécution des requêtes.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Requêtes Top N

Une requête Top `N` spécifie le classement du jeu de résultats et la limite du nombre de lignes renvoyées. Vous pouvez utiliser ce type de requête pour déterminer les valeurs Top `N` maximales ou Top `N` minimales pour vos jeux de données. Grâce à la poussée vers le bas Top `N`, le connecteur renvoie uniquement des lignes `N` classées à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le BigQuery connecteur Athena Google peut combiner ces expressions et les envoyer directement à Google BigQuery pour améliorer les fonctionnalités et réduire la quantité de données numérisées.

Les opérateurs de BigQuery connecteurs Athena Google suivants prennent en charge le transfert de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%')
ORDER BY col_a DESC
LIMIT 10;
```

Requêtes passthrough

Le BigQuery connecteur Google prend en charge les requêtes directes. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Google BigQuery, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant permet de transférer une requête vers une source de données dans Google BigQuery. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

Le projet Amazon Athena Google BigQuery Connector est concédé sous licence Apache-2.0.

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena Google Cloud Storage

Le connecteur Amazon Athena Google Cloud Storage permet à Amazon Athena d'exécuter des requêtes sur des fichiers Parquet et CSV stockés dans un compartiment Google Cloud Storage (GCS). Après avoir regroupé un ou plusieurs fichiers Parquet ou CSV dans un dossier partitionné ou non partitionné dans un compartiment GCS, vous pouvez les organiser dans une table de base de données AWS Glue.

Si Lake Formation est activé sur votre compte, le rôle IAM de votre connecteur Lambda fédéré Athena que vous avez déployé dans AWS Serverless Application Repository le doit avoir un accès en lecture dans Lake Formation au. AWS Glue Data Catalog

Pour consulter un article expliquant comment utiliser Athena pour exécuter des requêtes sur des fichiers Parquet ou CSV dans un bucket GCS, consultez le billet de blog consacré aux AWS mégadonnées Utiliser Amazon Athena pour interroger les données stockées dans Google Cloud Platform.

Prérequis

- Configurez une AWS Glue base de données et une table correspondant à votre bucket et à vos dossiers dans Google Cloud Storage. Pour consulter les étapes à suivre, reportez-vous à Configuration de bases de données et de tables dans AWS Glue plus loin dans ce document.
- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Actuellement, le connecteur ne prend en charge que le `VARCHAR` type des colonnes de partition (`string`ou `varchar` dans un schéma de AWS Glue table). Les autres types de champs de partition génèrent des erreurs lorsque vous les interrogez dans Athena.

Conditions

Les termes suivants se rapportent au connecteur GCS.

- Gestionnaire – Un gestionnaire Lambda qui accède à votre compartiment GCS. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre compartiment GCS.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre compartiment GCS.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre compartiment GCS.

Types de fichier pris en charge

Le connecteur GCS prend en charge les types de fichiers Parquet et CSV.

> **Note**
>
> Assurez-vous de ne pas placer les fichiers CSV et Parquet dans le même compartiment ou chemin GCS. Cela peut générer une erreur d'exécution lors d'une tentative de lecture des fichiers Parquet au format CSV ou vice versa.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur GCS.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.
- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.
- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.

- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.
- secret_manager_gcp_creds_name — Le nom du secret AWS Secrets Manager qui contient vos informations d'identification GCS au format JSON (par exemple,). `GoogleCloudPlatformCredentials`

Configuration de bases de données et de tables dans AWS Glue

La capacité d'inférence de schéma intégrée du connecteur GCS étant limitée, nous vous recommandons de l'utiliser AWS Glue pour vos métadonnées. Les procédures suivantes montrent comment créer une base de données et une table AWS Glue auxquelles vous pouvez accéder depuis Athena.

Création d'une base de données dans AWS Glue

Vous pouvez utiliser la AWS Glue console pour créer une base de données à utiliser avec le connecteur GCS.

Pour créer une base de données dans AWS Glue

1. Connectez-vous à la AWS Glue console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/glue/.
2. Dans le volet de navigation, sélectionnez Databases (Bases de données).
3. Choisissez Ajouter une base de données.
4. Dans le champ Name (Nom), saisissez le nom de la base de données que vous souhaitez utiliser avec le connecteur GCS.
5. Pour Emplacement, spécifiez `s3://google-cloud-storage-flag.` Cet emplacement indique au connecteur GCS que la AWS Glue base de données contient des tables pour les

données GCS à interroger dans Athena. Le connecteur reconnaît les bases de données qui présentent cet indicateur dans Athena et ignore les bases de données qui ne le présentent pas.

6. Choisissez Créer une base de données.

Création d'une table dans AWS Glue

Vous pouvez maintenant créer une table pour la base de données. Lorsque vous créez une AWS Glue table à utiliser avec le connecteur GCS, vous devez spécifier des métadonnées supplémentaires.

Pour créer une table dans la AWS Glue console

1. Dans le volet de navigation de la AWS Glue console, sélectionnez Tables.
2. Sur la page Tables, sélectionnez Add table (Ajouter une table).
3. Sur la page Set table properties (Définir les propriétés de la table), saisissez les informations suivantes.

   - Name (Nom) – Un nom unique pour la table.
   - Database (Base de données) – Sélectionnez la base de données AWS Glue que vous avez créée pour le connecteur GCS.
   - Include path (Inclure le chemin) – Dans la section Data store (Magasin de données), pour Include path (Inclure le chemin), saisissez l'emplacement d'URI pour GCS dont le préfixe est `gs://` (par exemple, `gs://gcs_table/data/`). Si vous disposez d'un ou de plusieurs dossiers de partition, ne les incluez pas dans le chemin.

     > **Note**
     >
     > Lorsque vous saisissez le chemin de table non `s3://`, la console AWS Glue affiche une erreur. Vous pouvez ignorer cette erreur. La création de la table réussira.

   - Data format (Format de données) – Pour Classification, sélectionnez CSV ou Parquet.
4. Choisissez Suivant.
5. Sur la page Choose or define schema (Choisir ou définir un schéma), il est vivement recommandé de définir un schéma de table, mais cela n'est pas obligatoire. Si vous ne définissez pas de schéma, le connecteur GCS tente d'en déduire un pour vous.

   Effectuez l'une des actions suivantes :

- Si vous souhaitez que le connecteur GCS tente de déduire un schéma pour vous, sélectionnez Next (Suivant), puis Create (Créer).
- Pour définir vous-même un schéma, suivez les étapes décrites dans la section suivante.

Définition d'un schéma de table dans AWS Glue

La définition d'un schéma de table dans AWS Glue nécessite davantage d'étapes, mais vous permet de mieux contrôler le processus de création de table.

Pour définir un schéma pour votre table dans AWS Glue

1. Sur la page Choose or define schema (Choisir ou définir un schéma), sélectionnez Add (Ajouter).
2. Dans la boîte de dialogue Add schema entry (Ajouter une entrée de schéma), saisissez un nom de colonne et un type de données.
3. Pour désigner la colonne en tant que colonne de partition, sélectionnez l’option Set as partition key (Définir en tant que clé de partition).
4. Sélectionnez Save (Enregistrer) pour enregistrer la colonne.
5. Sélectionnez Add (Ajouter) pour ajouter une autre colonne.
6. Lorsque vous avez terminé d’ajouter des colonnes, sélectionnez Next (Suivant).
7. Sur la page Review and create (Examiner et créer), examinez la table, puis sélectionnez Create (Créer).
8. Si votre schéma contient des informations de partition, suivez les étapes de la section suivante pour ajouter un modèle de partition aux propriétés de la table dans AWS Glue.

Ajout d'un modèle de partition aux propriétés de table dans AWS Glue

Si vos compartiments GCS comportent des partitions, vous devez ajouter le modèle de partition aux propriétés de la table dans AWS Glue.

Pour ajouter des informations de partition aux propriétés de la table AWS Glue

1. Sur la page de détails de la table que vous avez créée AWS Glue, choisissez Actions, Modifier la table.
2. Sur la page Edit table (Modifier le table), faites défiler l’écran vers le bas jusqu’à la section Table properties (Propriétés de la table).

3. Sélectionnez Add (Ajouter) pour ajouter une clé de partition.
4. Pour Key (Clé), saisissez **`partition.pattern`**. Cette clé définit le modèle de chemin de dossier.
5. Dans le champ Value (Valeur), saisissez un modèle de chemin de dossier tel que **`StateName=${statename}/ZipCode=${zipcode}/`**, **`statename`** et **`zipcode`** entourés par **`${}`** étant des noms de colonne de partition. Le connecteur GCS prend en charge les schémas de partition Hive et non Hive.
6. Lorsque vous avez terminé, choisissez Save.
7. Pour afficher les propriétés de la table que vous venez de créer, sélectionnez l'onglet Advanced properties (Propriétés avancées).

À ce stade, vous pouvez accéder à la console Athena. La base de données et la table que vous avez créées peuvent être consultées dans Athena. AWS Glue

Prise en charge du type de données

Les tables suivantes indiquent les types de données pris en charge pour les fichiers CSV et Parquet.

CSV

Nature des données	Type de données inféré
Les données ressemblent à un nombre	BIGINT
Les données ressemblent à une chaîne	VARCHAR
Les données ressemblent à une virgule flottante (flottante, double ou décimale)	DOUBLE
Les données ressemblent à une date	Horodatage
Données contenant des valeurs true/false (vrai/faux)	BOOL

Parquet

PARQUET	Athena (flèche)
BINAIRE	VARCHAR
BOOLEAN	BOOL
DOUBLE	DOUBLE
ENUM	VARCHAR
FIXED_LEN _BYTE_ARRAY	DECIMAL
FLOAT	FLOAT (32 bits)
INT32	1. INT32 2. DATEDAY (lorsque le type logique de la colonne Parquet est DATE)
INT64	1. INT64 2. TIMESTAMP (lorsque le type logique de la colonne Parquet est TIMESTAMP)
INT96	Horodatage
MAP	MAP
STRUCT	STRUCT
LIST	LIST

Autorisations nécessaires

Pour plus de détails sur les politiques IAM requises par ce connecteur, reportez-vous à la section `Policies` du fichier athena-gcs.yaml. La liste suivante résume les autorisations requises.

- Amazon S3 write access (Accès en écriture Amazon S3) – Le connecteur nécessite un accès en écriture à un emplacement dans Amazon S3 pour déverser les résultats à partir de requêtes volumineuses.

- Athena GetQueryExecution — Le connecteur utilise cette autorisation pour échouer rapidement lorsque la requête Athena en amont est terminée.
- AWS Glue Data Catalog— Le connecteur GCS nécessite un accès en lecture seule au pour AWS Glue Data Catalog obtenir des informations sur le schéma.
- CloudWatch Journaux : le connecteur a besoin d'accéder aux CloudWatch journaux pour stocker les journaux.

Performance

Lorsque le schéma de la table contient des champs de partition et que la propriété de la table `partition.pattern` est correctement configurée, vous pouvez inclure le champ de partition dans la clause `WHERE` de vos requêtes. Pour de telles requêtes, le connecteur GCS utilise les colonnes de partition pour affiner le chemin du dossier GCS et éviter d'analyser des fichiers inutiles dans les dossiers GCS.

Pour les jeux de données Parquet, la sélection d'un sous-ensemble de colonnes permet de réduire la quantité de données analysées. Cela se traduit généralement par un temps d'exécution des requêtes plus court lorsque la projection de colonnes est appliquée.

Pour les jeux de données CSV, la projection de colonnes n'est pas prise en charge et ne réduit pas la quantité de données analysées.

Les clauses `LIMIT` réduisent la quantité de données analysées, mais si vous ne fournissez pas de prédicat, vous devez vous attendre à ce que les requêtes `SELECT` dotées d'une clause `LIMIT` analysent au moins 16 Mo de données. Le connecteur GCS analyse une plus grande quantité de données pour les jeux de données plus volumineux que pour les jeux de données plus petits, indépendamment de la clause `LIMIT` appliquée. Par exemple, la requête `SELECT * LIMIT 10000` analyse une plus grande quantité de données pour un jeu de données sous-jacent plus volumineux que pour un jeu de données sous-jacent plus petit.

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour HBase

Le connecteur Amazon Athena HBase permet à Amazon Athena de communiquer avec vos instances Apache HBase afin que vous puissiez interroger vos données HBase avec SQL.

Contrairement aux magasins de données relatives traditionnels, les collections HBase n'ont pas de schéma défini. HBase n'a pas de magasin de métadonnées. Chaque entrée d'une collection HBase peut comporter des champs et des types de données différents.

Le connecteur HBase prend en charge deux mécanismes pour générer des informations de schéma de table : l'inférence de schéma de base et les AWS Glue Data Catalog métadonnées.

L'inférence de schéma est la valeur par défaut. Cette option analyse un petit nombre de documents de votre collection, réunit tous les champs et impose des champs dont les types de données ne se chevauchent pas. Cette option fonctionne bien pour les collections dont les entrées sont généralement uniformes.

Pour les collections comportant une plus grande variété de types de données, le connecteur prend en charge la récupération de métadonnées à partir du AWS Glue Data Catalog. Si le connecteur voit une AWS Glue base de données et une table qui correspondent à votre espace de noms HBase et aux noms de collection, il obtient ses informations de schéma à partir de la table correspondante AWS Glue . Lorsque vous créez votre AWS Glue table, nous vous recommandons d'en faire un sur-ensemble de tous les champs auxquels vous souhaitez accéder depuis votre collection HBase.

Si Lake Formation est activé sur votre compte, le rôle IAM de votre connecteur Lambda fédéré Athena que vous avez déployé dans AWS Serverless Application Repository le doit avoir un accès en lecture dans Lake Formation au. AWS Glue Data Catalog

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur HBase.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.

- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.
- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.
- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.
- disable_glue — (Facultatif) S'il est présent et défini sur true, le connecteur ne tente pas de récupérer des métadonnées supplémentaires à partir de. AWS Glue
- glue_catalog – (Facultatif) Utilisez cette option pour spécifier un catalogue AWS Glue entre compte. Par défaut, le connecteur tente d'obtenir des métadonnées à partir de son propre AWS Glue compte.
- default_hbase – S'il est présent, il spécifie une chaîne de connexion HBase à utiliser lorsqu'aucune variable d'environnement spécifique au catalogue n'existe.
- enable_case_insensitive_match — (Facultatif) Lorsque, effectue des recherches sans distinction majuscules/minuscules sur les noms de tables dans `true` HBase. L'argument par défaut est `false`. À utiliser si votre requête contient des noms de table en majuscules.

Définition des chaînes de connexion

Vous pouvez fournir une ou plusieurs propriétés qui définissent les détails de connexion HBase pour les instances HBase que vous utilisez avec le connecteur. Pour ce faire, définissez une variable d'environnement Lambda qui correspond au nom du catalogue que vous souhaitez utiliser dans

Athena. Supposons, par exemple, que vous souhaitiez utiliser les requêtes suivantes pour interroger deux instances HBase différentes depuis Athena :

```
SELECT * FROM "hbase_instance_1".database.table
```

```
SELECT * FROM "hbase_instance_2".database.table
```

Avant de pouvoir utiliser ces deux instructions SQL, vous devez ajouter deux variables d'environnement à votre fonction Lambda : `hbase_instance_1` et `hbase_instance_2`. La valeur pour chacune d'elles doit être une chaîne de connexion HBase au format suivant :

```
master_hostname:hbase_port:zookeeper_port
```

Utilisation de secrets

Vous pouvez éventuellement utiliser AWS Secrets Manager une partie ou la totalité de la valeur pour les détails de votre chaîne de connexion. Pour utiliser la fonctionnalité de requête fédérée d'Athena avec Secrets Manager, le VPC connecté à votre fonction Lambda doit avoir un accès internet ou un point de terminaison de VPC pour vous connecter à Secrets Manager.

Si vous utilisez la syntaxe `${my_secret}` pour mettre le nom d'un secret provenant de Secrets Manager dans votre chaîne de connexion, le connecteur remplace le nom secret par vos valeurs de nom d'utilisateur et de mot de passe provenant de Secrets Manager.

Supposons, par exemple, que vous définissiez la variable d'environnement Lambda pour `hbase_instance_1` sur la valeur suivante :

```
${hbase_host_1}:${hbase_master_port_1}:${hbase_zookeeper_port_1}
```

Le SDK Athena Query Federation tente automatiquement de récupérer un secret nommé `hbase_instance_1_creds` à partir de Secrets Manager et d'injecte cette valeur à la place de `${hbase_instance_1_creds}`. Toute partie de la chaîne de connexion qui est entourée par la combinaison de caractères `${ }` est interprétée comme un secret de Secrets Manager. Si vous spécifiez un nom de secret que le connecteur ne trouve pas dans Secrets Manager, le connecteur ne remplace pas le texte.

Configuration de bases de données et de tables dans AWS Glue

L’inférence de schéma intégrée du connecteur prend en charge uniquement les valeurs sérialisées dans HBase sous forme de chaînes (par exemple, `String.valueOf(int)`). La capacité d’inférence de schéma intégrée du connecteur étant limitée, vous pouvez souhaiter utiliser plutôt AWS Glue pour les métadonnées. Pour activer une AWS Glue table à utiliser avec HBase, vous devez disposer d'une AWS Glue base de données et d'une table dont les noms correspondent à l'espace de noms HBase et à la table pour lesquels vous souhaitez fournir des métadonnées supplémentaires. L’utilisation des conventions de dénomination des familles de colonnes HBase est facultative, mais pas obligatoire.

Pour utiliser une AWS Glue table pour des métadonnées supplémentaires

1. Lorsque vous modifiez la table et la base de données dans la AWS Glue console, ajoutez les propriétés de table suivantes :

   - hbase-metadata-flag— Cette propriété indique au connecteur HBase que le connecteur peut utiliser la table pour des métadonnées supplémentaires. Vous pouvez fournir n’importe quelle valeur pour `hbase-metadata-flag` tant que la propriété `hbase-metadata-flag` est présente dans la liste des propriétés de la table.
   - hbase-native-storage-flag— Utilisez cet indicateur pour activer ou désactiver les deux modes de sérialisation de valeurs pris en charge par le connecteur. Par défaut, lorsque ce champ n’est pas présent, le connecteur suppose que toutes les valeurs sont stockées dans HBase sous forme de chaînes. En tant que tel, il tentera d’analyser les types de données tels que `INT`, `BIGINT` et `DOUBLE` à partir de HBase sous forme de chaînes. Si ce champ est défini avec une valeur quelconque de la table AWS Glue, le connecteur passe en mode de stockage « natif » et tente de lire`INT`, `BIGINTBIT`, et `DOUBLE` sous forme d'octets à l'aide des fonctions suivantes :

     ```
     ByteBuffer.wrap(value).getInt()
     ByteBuffer.wrap(value).getLong()
     ByteBuffer.wrap(value).get()
     ByteBuffer.wrap(value).getDouble()
     ```

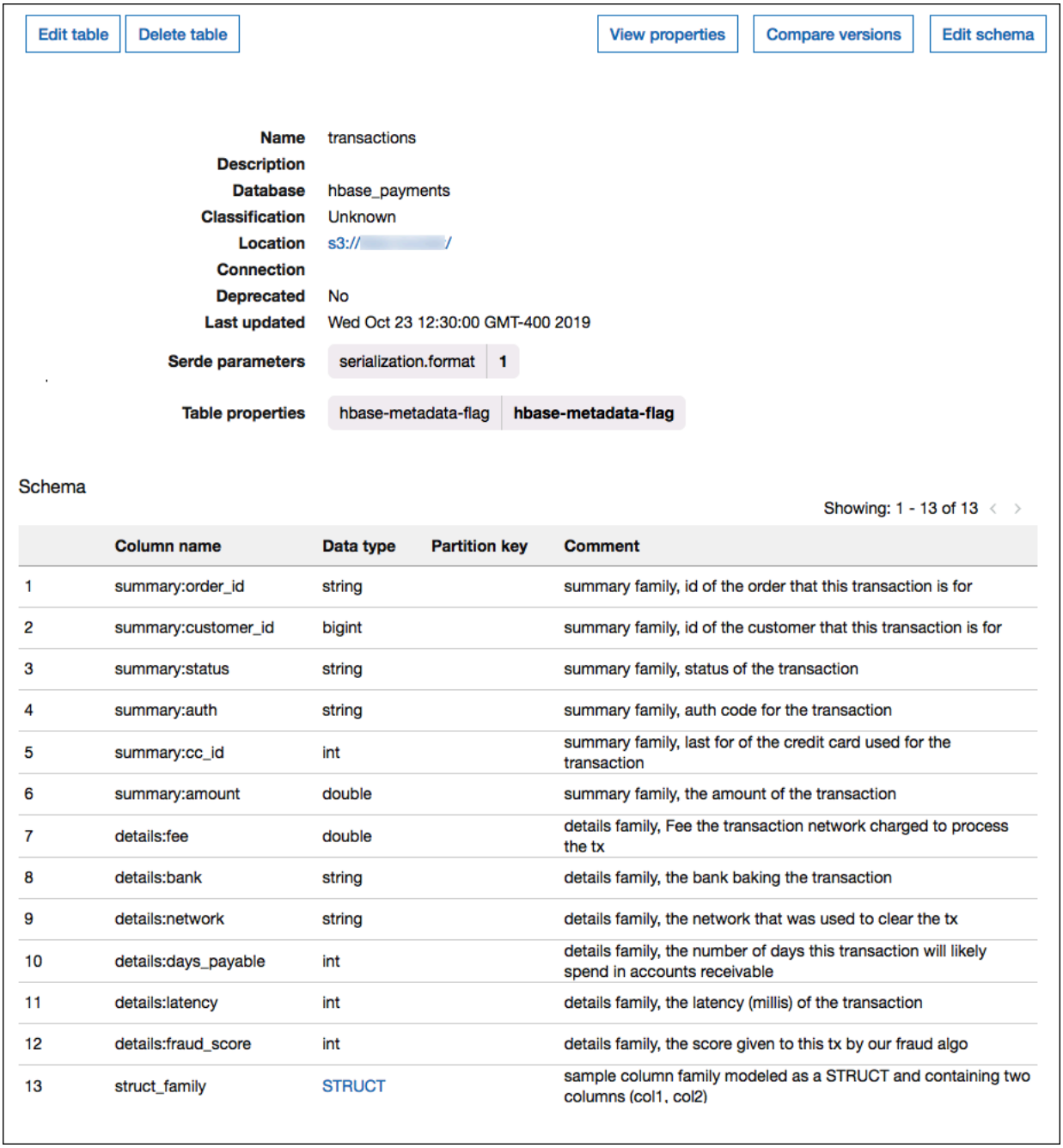
2. Assurez-vous d'utiliser les types de données appropriés AWS Glue tels que listés dans ce document.

Modélisation de familles de colonnes

Le connecteur Athena HBase permet de modéliser les familles de colonnes HBase de deux manières : des noms entièrement qualifiés (aplatis) tels que `family:column`, ou à l'aide d'objets `STRUCT`.

Dans le modèle `STRUCT`, le nom du champ `STRUCT` doit correspondre à la famille de colonnes, tandis que les enfants de `STRUCT` doivent correspondre aux noms des colonnes de la famille. Cependant, étant donné que les lectures de prédicats poussés vers le bas et en colonnes ne sont pas encore totalement prises en charge pour les types complexes tels que `STRUCT`, l'utilisation de `STRUCT` n'est actuellement pas conseillée.

L'image suivante montre une table configurée dans AWS Glue qui utilise une combinaison des deux approches.



	Column name	Data type	Partition key	Comment
1	summary:order_id	string		summary family, id of the order that this transaction is for
2	summary:customer_id	bigint		summary family, id of the customer that this transaction is for
3	summary:status	string		summary family, status of the transaction
4	summary:auth	string		summary family, auth code for the transaction
5	summary:cc_id	int		summary family, last for of the credit card used for the transaction
6	summary:amount	double		summary family, the amount of the transaction
7	details:fee	double		details family, Fee the transaction network charged to process the tx
8	details:bank	string		details family, the bank baking the transaction
9	details:network	string		details family, the network that was used to clear the tx
10	details:days_payable	int		details family, the number of days this transaction will likely spend in accounts receivable
11	details:latency	int		details family, the latency (millis) of the transaction
12	details:fraud_score	int		details family, the score given to this tx by our fraud algo
13	struct_family	STRUCT		sample column family modeled as a STRUCT and containing two columns (col1, col2)

Prise en charge du type de données

Le connecteur extrait toutes les valeurs HBase en tant que type d'octet de base. Ensuite, en fonction de la façon dont vous avez défini vos tables dans AWS Glue Data Catalog, il mappe les valeurs dans l'un des types de données Apache Arrow du tableau suivant.

AWS Glue type de données	Type de données Apache Arrow
int	INT
bigint	BIGINT
double	FLOAT8
float	FLOAT4
boolean	BIT
binary	VARBINARY
chaîne	VARCHAR

> **Note**
>
> Si vous ne l'utilisez pas AWS Glue pour compléter vos métadonnées, l'inférence du schéma du connecteur utilise uniquement les types de données `BIGINT`FLOAT8, et`VARCHAR`.

Autorisations nécessaires

Pour plus de détails sur les politiques IAM requises par ce connecteur, reportez-vous à la section `Policies` du fichier athena-hbase.yaml. La liste suivante résume les autorisations requises.

- Amazon S3 write access (Accès en écriture Amazon S3) – Le connecteur nécessite un accès en écriture à un emplacement dans Amazon S3 pour déverser les résultats à partir de requêtes volumineuses.
- Athena GetQueryExecution — Le connecteur utilise cette autorisation pour échouer rapidement lorsque la requête Athena en amont est terminée.
- AWS Glue Data Catalog— Le connecteur HBase nécessite un accès en lecture seule au pour AWS Glue Data Catalog obtenir des informations sur le schéma.
- CloudWatch Journaux : le connecteur a besoin d'accéder aux CloudWatch journaux pour stocker les journaux.

- AWS Secrets Manager accès en lecture — Si vous choisissez de stocker les détails du point de terminaison HBase dans Secrets Manager, vous devez autoriser le connecteur à accéder à ces secrets.
- Accès VPC – Le connecteur nécessite la possibilité d'attacher des interfaces à votre VPC et de les détacher afin qu'il puisse s'y connecter et communiquer avec vos instances HBase.

Performance

Le connecteur Athena HBase tente de paralléliser les requêtes sur votre instance HBase en lisant chaque serveur de région en parallèle. Le connecteur Athena HBase effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête.

La fonction Lambda effectue également une poussée vers le bas de projection pour réduire les données analysées par la requête. Cependant, la sélection d'un sous-ensemble de colonnes entraîne parfois un temps d'exécution plus long de la requête. Les clauses `LIMIT` réduisent la quantité de données analysées, mais si vous ne fournissez pas de prédicat, vous devez vous attendre à ce que les requêtes `SELECT` avec une clause `LIMIT` analysent au moins 16 Mo de données.

HBase est susceptible de faire échouer des requêtes et d'avoir des temps d'exécution variables. Vous devrez peut-être réessayer vos requêtes plusieurs fois pour qu'elles aboutissent. Le connecteur HBase résiste à la limitation due à la simultanéité.

Requêtes passthrough

Le connecteur HBase prend en charge les requêtes passthrough et est basé sur NoSQL. Pour plus d'informations sur l'interrogation d'Apache HBase à l'aide du filtrage, consultez la section Langage des filtres dans la documentation d'Apache.

Pour utiliser des requêtes passthrough avec HBase, utilisez la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            database => 'database_name',
            collection => 'collection_name',
            filter => '{query_syntax}'
        ))
```

L'exemple suivant permet de filtrer les requêtes HBase passthrough pour les employés âgés de 24 ou 30 ans au sein de la `employee` collection de la `default` base de données.

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'default',
            COLLECTION => 'employee',
            FILTER => 'SingleColumnValueFilter(''personaldata'', ''age'', =,
 ''binary:30'')' ||
                      ' OR SingleColumnValueFilter(''personaldata'', ''age'', =,
 ''binary:24'')'
        ))
```

Informations de licence

Le projet de connecteur HBase Amazon Athena est concédé sous licence dans le cadre de la licence Apache-2.0.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Hortonworks

Le connecteur Amazon Athena pour Hortonworks permet à Amazon Athena d'exécuter des requêtes SQL sur la plateforme de données Cloudera Hortonworks. Le connecteur transforme vos requêtes Athena SQL en syntaxe HiveQL équivalente.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .

Conditions

Les termes suivants se rapportent au connecteur Hortonworks Hive.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Hortonworks Hive.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
hive://${jdbc_connection_string}
```

Utilisation d’un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l’aide d’une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`HiveMuxCompositeHandler`
Gestionnaire de métadonnées	`HiveMuxMetadataHandler`
Gestionnaire d’enregis trements	`HiveMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
$*catalog*_connecti on_string	Obligatoire. Chaîne de connexion d’instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d’Athena est `myhivecatalog` , le nom de la variable d’environnement est alors `myhivecatalog_connection_st ring` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *AWS_LAMBD A_FUNCTION_NAME* `}`.

Les exemples de propriétés suivants concernent une fonction Hive MUX Lambda qui prend en charge deux instances de base de données :`hive1` (par défaut) et `hive2`.

Propriété	Valeur
`default`	`hive://jdbc:hive2://hive1:10000/default?${Test/RDS/hive1}`
`hive_catalog1_connection_string`	`hive://jdbc:hive2://hive1:10000/default?${Test/RDS/hive1}`
`hive_catalog2_connection_string`	`hive://jdbc:hive2://hive2:10000/default?UID=sample&PWD=sample`

Fourniture des informations d'identification

Pour fournir un nom d'utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

  Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS

Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${Test/RDS/hive1host}`.

```
hive://jdbc:hive2://hive1host:10000/default?...&${Test/RDS/hive1host}&...
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
hive://jdbc:hive2://hive1host:10000/default?...&UID=sample2&PWD=sample2&...
```

Actuellement, le connecteur Hortonworks Hive reconnaît les propriétés JDBC `UID` et `PWD`.

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance Hortonworks Hive.

Type de gestionnaire	Classe
Gestionnaire de composites	`HiveCompositeHandler`
Gestionnaire de métadonnées	`HiveMetadataHandler`
Gestionnaire d'enregistrements	`HiveRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une instance Hortonworks Hive unique prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`hive://jdbc:hive2://hive1host:10000/default?secret=${Test/RDS/hive1host}`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant montre les types de données correspondants pour JDBC, Hortonworks Hive et Arrow.

JDBC	Hortonworks Hive	Flèche
Booléen	Booléen	Bit
Entier	TINYINT	Tiny
Court	SMALLINT	Smallint
Entier	INT	Int
Long	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8
Date	date	DateDay
Horodatage	timestamp	DateMilli
Chaîne	VARCHAR	Varchar
Octets	octets	Varbinary
BigDecimal	Décimal	Décimal
ARRAY	N/A (voir la remarque)	Liste

Note

Hortonworks Hive ne prend actuellement pas en charge les types d'agrégat `ARRAY`, `MAP`, `STRUCT` ou `UNIONTYPE`. Les colonnes de types d'agrégats sont traitées comme des colonnes `VARCHAR` dans SQL.

Partitions et déversements

Les partitions sont utilisées pour déterminer comment générer des divisions pour le connecteur. Athena construit une colonne synthétique de type `varchar` qui représente le schéma de partitionnement de la table afin d'aider le connecteur à générer des divisions. Le connecteur ne modifie pas la définition réelle de la table.

Performance

Hortonworks Hive prend en charge les partitions statiques. Le connecteur Athena Hortonworks Hive peut récupérer les données de ces partitions en parallèle. Si vous souhaitez interroger des jeux de données très volumineux avec une distribution de partition uniforme, le partitionnement statique est fortement recommandé. La sélection d'un sous-ensemble de colonnes accélère considérablement l'exécution des requêtes et réduit le nombre de données analysées. Le connecteur Hortonworks Hive résiste à la limitation due à la simultanéité.

Le connecteur Athena Hortonworkds Hive effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Les clauses `LIMIT`, les prédicats simples et les expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution des requêtes.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena Hortonworks Hive peut combiner ces expressions et les pousser directement vers Cloudera Hive pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena Hortonworks Hive suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE

- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%')
LIMIT 10;
```

Requêtes directes

Le connecteur Hortonworks Hive prend en charge les requêtes directes. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Hortonworks Hive, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Hortonworks Hive. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les informations les plus récentes sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur Hortonworks Hive sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena Apache Kafka

Le connecteur Amazon Athena pour Apache Kafka permet à Amazon Athena d'exécuter des requêtes SQL sur vos rubriques Apache Kafka. Utilisez ce connecteur pour afficher les rubriques Apache Kafka sous forme de tableaux et les messages sous forme de lignes dans Athena.

Prérequis

Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Les types de données de date et d'horodatage dans des conditions de filtre doivent être convertis en types de données appropriés.
- Les types de données date et horodatage ne sont pas pris en charge pour le type de fichier CSV et sont traités comme des valeurs varchar.
- Le mappage dans des champs JSON imbriqués n'est pas pris en charge. Le connecteur ne mappe que les champs de niveau supérieur.
- Le connecteur ne prend pas en charge les types complexes. Les types complexes sont interprétés comme des chaînes.
- Pour extraire ou travailler avec des valeurs JSON complexes, utilisez les fonctions relatives à JSON disponibles dans Athena. Pour plus d'informations, consultez Extraction de données JSON à partir de chaînes.
- Le connecteur ne prend pas en charge l'accès aux métadonnées des messages Kafka.

Conditions

- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Point de terminaison Kafka – Chaîne de texte qui établit une connexion à une instance Kafka.

Compatibilité des clusters

Le connecteur Kafka peut être utilisé avec les types de clusters suivants.

- Kafka autonome – Connexion directe à Kafka (authentifiée ou non).
- Confluent – Une connexion directe à Confluent Kafka. Pour plus d'informations sur l'utilisation d'Athena avec les données Confluent Kafka, consultez Visualiser les données Confluent sur Amazon à l'aide d' QuickSight Amazon Athena sur le blog Business Intelligence.AWS

Connexion à Confluent

La connexion à Confluent nécessite les étapes suivantes :

1. Générer une clé d'API à partir de Confluent.
2. Enregistrer le nom d'utilisateur et le mot de passe de la clé API Confluent dans AWS Secrets Manager.
3. Indiquer le nom secret de la variable d'environnement `secrets_manager_secret` dans le connecteur Kafka.
4. Suivez les étapes décrites dans la section Configuration du connecteur Kafka de ce document.

Méthodes d'authentification prises en charge

Le connecteur prend en charge les méthodes d'authentification suivantes.

- SSL
- SASL/SCRAM

- SASL/PLAIN
- SASL/PLAINTEXT
- NO_AUTH
- Plateforme Kafka et Confluent autogérée – SSL, SASL/SCRAM, SASL/PLAINTEXT, NO_AUTH
- Cloud Kafka et Confluent autogéré – SASL/PLAIN

Pour plus d'informations, consultez Configuration de l'authentification pour le connecteur Athena Kafka.

Formats de données d'entrée pris en charge

Le connecteur prend en charge les formats de données d'entrée suivants.

- JSON
- CSV

Paramètres

Utilisez les variables d'environnement Lambda mentionnées dans cette rubrique pour configurer le connecteur Kafka d'Athena.

- auth_type – Spécifie le type d'authentification du cluster. Le connecteur prend en charge les types d'authentification suivants :
  - NO_AUTH – Connectez-vous directement à Kafka (par exemple, à un cluster Kafka déployé sur une instance EC2 qui n'utilise pas l'authentification).
  - SASL_SSL_PLAIN – Cette méthode utilise le protocole de sécurité `SASL_SSL` et le mécanisme SASL `PLAIN`. Pour plus d'informations, consultez la page Configuration SASL dans la documentation Apache Kafka.
  - SASL_PLAINTEXT_PLAIN – Cette méthode utilise le protocole de sécurité `SASL_PLAINTEXT` et le mécanisme SASL `PLAIN`. Pour plus d'informations, consultez la page Configuration SASL dans la documentation Apache Kafka.
  - SASL_SSL_SCRAM_SHA512 – Vous pouvez utiliser ce type d'authentification pour contrôler l'accès à vos clusters Apache Kafka. Cette méthode enregistre le nom d'utilisateur et le mot de passe dans AWS Secrets Manager. Le secret doit être associé au cluster Kafka. Pour plus d'informations, consultez Authentification à l'aide de SASL/SCRAM dans la documentation Apache Kafka.

- SASL_PLAINTEXT_SCRAM_SHA512 – Cette méthode utilise le protocole de sécurité `SASL_PLAINTEXT` et le mécanisme `SCRAM_SHA512 SASL`. Cette méthode utilise votre nom d'utilisateur et votre mot de passe enregistrés dans AWS Secrets Manager. Pour plus d'informations, consultez la section Configuration SASL dans la documentation Apache Kafka.
- SSL – L'authentification SSL utilise des fichiers keystore et truststore pour se connecter au cluster Apache Kafka. Vous devez générer les fichiers keystore et truststore, les charger dans un compartiment Amazon S3 et fournir la référence à Amazon S3 lorsque vous déployez le connecteur. Le magasin de clés, le magasin de confiance et la clé SSL sont stockés dans AWS Secrets Manager. Votre client doit fournir la clé AWS secrète lors du déploiement du connecteur. Pour plus d'informations, consultez Chiffrement et authentification à l'aide de SSL dans la documentation Apache Kafka.

  Pour plus d'informations, consultez Configuration de l'authentification pour le connecteur Athena Kafka.

- certificates_s3_reference – L'emplacement Amazon S3 qui contient les certificats (les fichiers keystore et truststore).
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.
- kafka_endpoint – Les détails du point de terminaison à fournir à Kafka.
- secrets_manager_secret – Le nom du secret AWS dans lequel sont enregistrées les informations d'identification.
- Paramètres de déversement – Les fonctions Lambda stockent temporairement (« déversent ») les données qui ne tiennent pas en mémoire vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement. Utilisez les paramètres du tableau suivant pour spécifier l'emplacement de déversement.

Paramètre	Description
`spill_bucket`	Obligatoire. Le nom du compartiment Amazon S3 où la fonction Lambda peut déverser des données.

Paramètre	Description
`spill_prefix`	Obligatoire. Le préfixe dans le compartiment de déversement où la fonction Lambda peut déverser des données.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

- ID du sous-réseau – Un ou plusieurs ID de sous-réseau qui correspondent au sous-réseau que la fonction Lambda peut utiliser pour accéder à votre source de données .
  - Cluster Kafka public ou cluster Confluent Cloud standard – Associez le connecteur à un sous-réseau privé doté d'une passerelle NAT.
  - Cluster Confluent Cloud avec connectivité privée – Associez le connecteur à un sous-réseau privé qui possède une route vers le cluster Confluent Cloud.
    - Pour AWS Transit Gateway, les sous-réseaux doivent se trouver dans un VPC attaché à la même passerelle de transit qu'utilise Confluent Cloud.
    - Pour l'appairage de VPC, les sous-réseaux doivent se trouver dans un VPC appairé au VPC Confluent Cloud.
    - Pour AWS PrivateLink, les sous-réseaux doivent se trouver dans un VPC qui possède une route vers les points de terminaison du VPC qui se connectent à Confluent Cloud.

> **Note**
>
> Si vous déployez le connecteur dans un VPC afin d'accéder à des ressources privées et si vous souhaitez également vous connecter à un service accessible au public tel que Confluent, vous devez associer le connecteur à un sous-réseau privé doté d'une passerelle NAT. Pour plus d'informations, consultez Passerelles NAT dans le Guide de l'utilisateur Amazon VPC.

Prise en charge du type de données

Le tableau suivant indique les types de données correspondants pris en charge par Kafka et Apache Arrow.

Kafka	Flèche
CHAR	VARCHAR
VARCHAR	VARCHAR
TIMESTAMP	MILLISECOND
DATE	DAY
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	FLOAT8
DOUBLE	FLOAT8

Partitions et déversements

Les rubriques Kafka sont divisées en partitions. Chaque partition est ordonnée. Chaque message dans une partition a un ID incrémentiel appelé offset. Chaque partition Kafka est ensuite divisée en plusieurs parties pour un traitement parallèle. Les données sont disponibles pour la période de rétention configurée dans les clusters Kafka.

Bonnes pratiques

Il est recommandé d'utiliser la poussée vers le bas des prédicats lorsque vous interrogez Athena, comme dans les exemples suivants.

```
SELECT *
FROM "kafka_catalog_name"."glue_schema_registry_name"."glue_schema_name"
```

```
WHERE integercol = 2147483647
```

```
SELECT *
FROM "kafka_catalog_name"."glue_schema_registry_name"."glue_schema_name"
WHERE timestampcol >= TIMESTAMP '2018-03-25 07:30:58.878'
```

Configuration du connecteur Kafka

Pour pouvoir utiliser le connecteur, vous devez configurer votre cluster Apache Kafka, utilisez le Registre de schémas AWS Glue pour définir votre schéma et configurez l'authentification pour le connecteur.

Lorsque vous travaillez avec le registre des AWS Glue schémas, tenez compte des points suivants :

- Assurez-vous que le texte du champ Description du registre de schémas AWS Glue inclut la chaîne `{AthenaFederationKafka}`. Cette chaîne de marquage est obligatoire pour les AWS Glue registres que vous utilisez avec le connecteur Amazon Athena Kafka.
- Pour des performances optimales, n'utilisez que des minuscules pour vos noms de bases de données et de tables. L'utilisation d'une casse mixte oblige le connecteur à effectuer une recherche insensible à la casse, ce qui demande plus de temps de calcul.

Pour configurer votre environnement Apache Kafka et votre registre de AWS Glue schémas

1. Configurez votre environnement Apache Kafka.
2. Téléchargez le fichier de description de la rubrique Kafka (c'est-à-dire son schéma) au format JSON dans le registre des AWS Glue schémas. Pour plus d'informations, consultez la section Intégration à AWS Glue Schema Registry dans le guide du AWS Glue développeur. Pour des exemples de schémas, voir la rubrique suivante.

Exemples de schémas pour le registre de schémas AWS Glue

Utilisez le format des exemples de cette rubrique lorsque vous téléchargez votre schéma dans le registre de schémas AWS Glue.

Exemple de schéma de type JSON

Dans l'exemple suivant, le schéma à créer dans le registre des AWS Glue `json` schémas indique la valeur pour `dataFormat` et les utilisations `datatypejson` pour`topicName`.

> **Note**
>
> La valeur de `topicName` doit utiliser la même casse que le nom de la rubrique de Kafka.

```
{
  "topicName": "datatypejson",
  "message": {
    "dataFormat": "json",
    "fields": [
      {
        "name": "intcol",
        "mapping": "intcol",
        "type": "INTEGER"
      },
      {
        "name": "varcharcol",
        "mapping": "varcharcol",
        "type": "VARCHAR"
      },
      {
        "name": "booleancol",
        "mapping": "booleancol",
        "type": "BOOLEAN"
      },
      {
        "name": "bigintcol",
        "mapping": "bigintcol",
        "type": "BIGINT"
      },
      {
        "name": "doublecol",
        "mapping": "doublecol",
        "type": "DOUBLE"
      },
      {
        "name": "smallintcol",
        "mapping": "smallintcol",
        "type": "SMALLINT"
      },
      {
        "name": "tinyintcol",
```

```
        "mapping": "tinyintcol",
        "type": "TINYINT"
      },
      {
        "name": "datecol",
        "mapping": "datecol",
        "type": "DATE",
        "formatHint": "yyyy-MM-dd"
      },
      {
        "name": "timestampcol",
        "mapping": "timestampcol",
        "type": "TIMESTAMP",
        "formatHint": "yyyy-MM-dd HH:mm:ss.SSS"
      }
    ]
  }
}
```

Exemple de schéma de type CSV

Dans l'exemple suivant, le schéma à créer dans le registre des AWS Glue `csv` schémas indique la valeur pour `dataFormat` et les utilisations `datatypecsvbulk` pour`topicName`. La valeur de `topicName` doit utiliser la même casse que le nom de la rubrique de Kafka.

```
{
  "topicName": "datatypecsvbulk",
  "message": {
    "dataFormat": "csv",
    "fields": [
      {
        "name": "intcol",
        "type": "INTEGER",
        "mapping": "0"
      },
      {
        "name": "varcharcol",
        "type": "VARCHAR",
        "mapping": "1"
      },
      {
        "name": "booleancol",
        "type": "BOOLEAN",
```

```
        "mapping": "2"
      },
      {
        "name": "bigintcol",
        "type": "BIGINT",
        "mapping": "3"
      },
      {
        "name": "doublecol",
        "type": "DOUBLE",
        "mapping": "4"
      },
      {
        "name": "smallintcol",
        "type": "SMALLINT",
        "mapping": "5"
      },
      {
        "name": "tinyintcol",
        "type": "TINYINT",
        "mapping": "6"
      },
      {
        "name": "floatcol",
        "type": "DOUBLE",
        "mapping": "7"
      }
    ]
  }
}
```

Configuration de l'authentification pour le connecteur Athena Kafka

Vous pouvez utiliser diverses méthodes pour vous authentifier dans votre cluster Apache Kafka, notamment SSL, SASL/SCRAM, SASL/PLAIN et SASL/PLAINTEXT.

Le tableau suivant présente les types d'authentification pour le connecteur ainsi que le protocole de sécurité et le mécanisme SASL pour chacun d'entre eux. Pour de plus amples informations, consultez la section Sécurité de la documentation Apache Kafka.

auth_type	security.protocol	sasl.mechanism	Compatibilité avec les types de cluster
SASL_SSL_PLAIN	SASL_SSL	PLAIN	• Kafka autogéré • Plateforme Confluent • Cloud confluent
SASL_PLAINTEXT_PLAIN	SASL_PLAINTEXT	PLAIN	• Kafka autogéré • Plateforme Confluent
SASL_SSL_SCRAM_SHA512	SASL_SSL	SCRAM-SHA-512	• Kafka autogéré • Plateforme Confluent
SASL_PLAINTEXT_SCRAM_SHA512	SASL_PLAINTEXT	SCRAM-SHA-512	• Kafka autogéré • Plateforme Confluent
SSL	SSL	N/A	• Kafka autogéré • Plateforme Confluent

SSL

Si le cluster est authentifié SSL, vous devez générer les fichiers truststore (magasin d'approbations) et keystore (magasin de clés) et les télécharger dans le compartiment Amazon S3. Vous devez fournir cette référence Amazon S3 lorsque vous déployez le connecteur. Les clés keystore, truststore et SSL sont stockées dans AWS Secrets Manager. Vous fournissez la clé AWS secrète lorsque vous déployez le connecteur.

Pour plus d'informations sur la création d'un secret dans Secrets Manager, voir la rubrique Créer un secret AWS Secrets Manager.

Pour utiliser ce type d'authentification, définissez les variables d'environnement comme indiqué dans le tableau suivant.

Paramètre	Valeur
auth_type	SSL

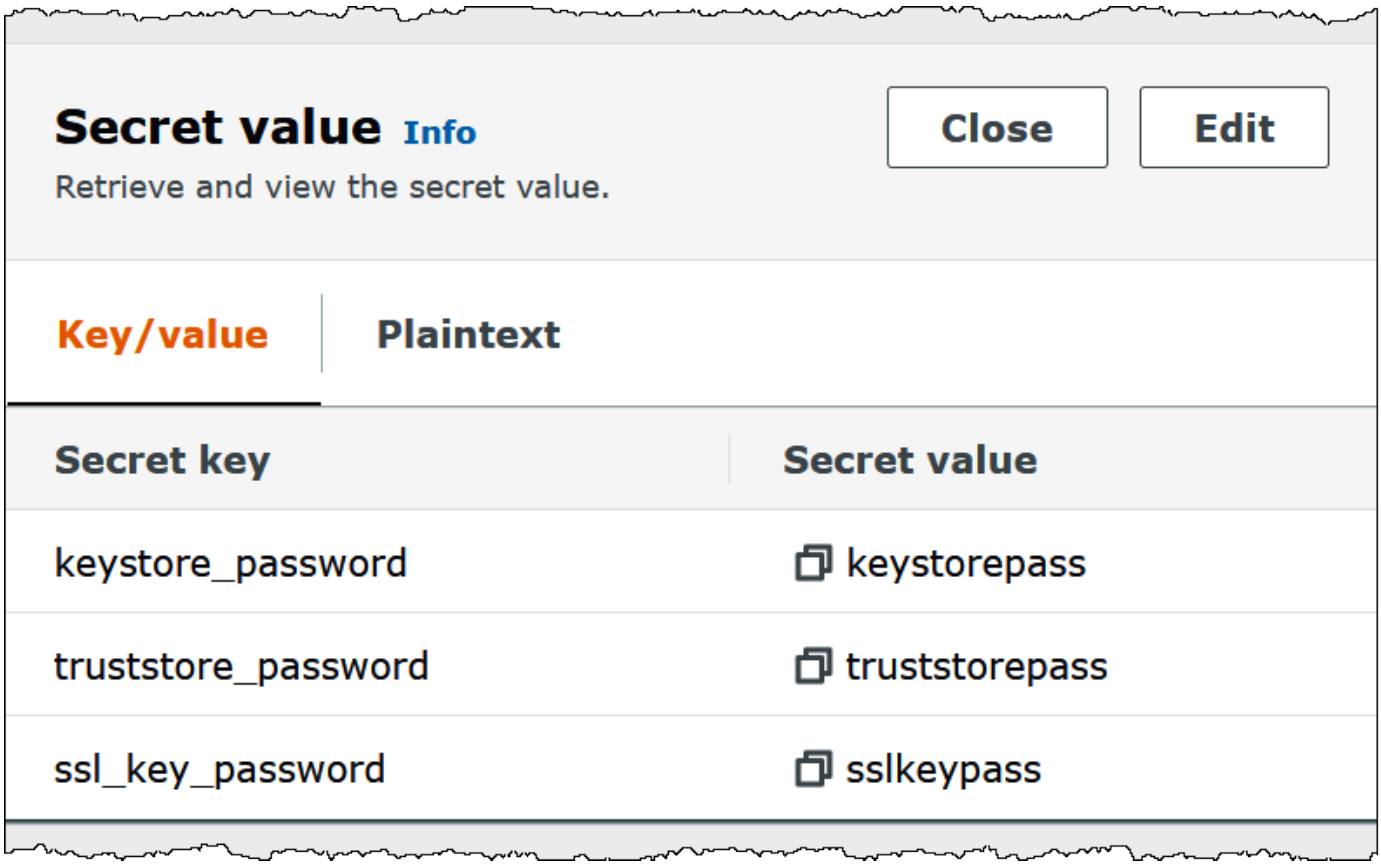
Paramètre	Valeur
`certificates_s3_reference`	L'emplacement Amazon S3 qui contient les certificats.
`secrets_manager_secret`	Le nom de votre clé AWS secrète.

Après avoir créé un secret dans Secrets Manager, vous pouvez le consulter dans la console Secrets Manager.

Visualisation de votre secret dans Secrets Manager

1. Ouvrez la console Secrets Manager en suivant le lien https://console.aws.amazon.com/secretsmanager/.
2. Dans le volet de navigation, sélectionnez Secrets.
3. Sur la page Secrets, choisissez le lien vers votre secret.
4. Sur la page de détails de votre secret, sélectionnez Retrieve secret value (Récupérer la valeur du secret).

   L'image suivante montre un exemple de secret avec trois paires clé/valeur : `keystore_password`, `truststore_password` et `ssl_key_password`.



Pour plus d'informations sur l'utilisation de SSL avec Kafka, consultez Chiffrement et authentification à l'aide de SSL dans la documentation Apache Kafka.

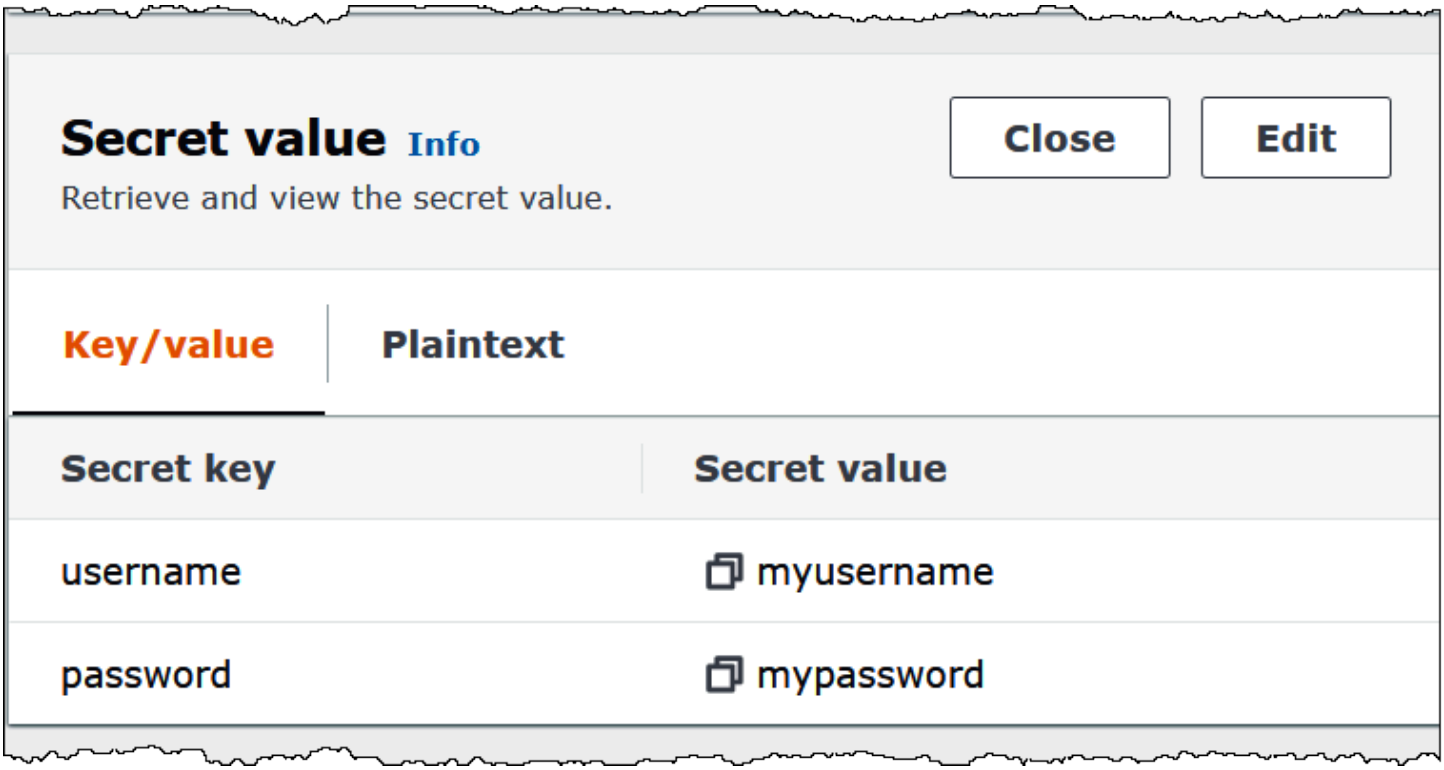
SASL/SCRAM

Si votre cluster utilise l'authentification SCRAM, fournissez la clé Secrets Manager qui est associée au cluster lorsque vous déployez le connecteur. Les informations d'identification AWS de l'utilisateur (clé secrète et clé d'accès) sont utilisées pour s'authentifier au cluster.

Définissez les variables d'environnement comme indiqué dans le tableau suivant.

Paramètre	Valeur
`auth_type`	`SASL_SSL_SCRAM_SHA512`
`secrets_manager_secret`	Le nom de votre clé AWS secrète.

L'image suivante montre un exemple de secret dans la console Secrets Manager avec deux paires clé/valeur : une pour `username` et une pour `password`.



Pour plus d'informations sur l'utilisation de SASL/SCRAM avec Kafka, consultez Authentification à l'aide de SASL/SCRAM dans la documentation Apache Kafka.

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour MSK

Le connecteur Amazon Athena pour Amazon MSK permet à Amazon Athena d'exécuter des requêtes SQL sur vos rubriques Apache Kafka. Utilisez ce connecteur pour afficher les rubriques Apache Kafka sous forme de tableaux et les messages sous forme de lignes dans Athena. Pour plus d'informations, consultez Analyser les données de streaming en temps réel dans Amazon MSK avec Amazon Athena sur AWS le blog Big Data.

Prérequis

Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Les types de données de date et d'horodatage dans des conditions de filtre doivent être convertis en types de données appropriés.
- Les types de données date et horodatage ne sont pas pris en charge pour le type de fichier CSV et sont traités comme des valeurs varchar.
- Le mappage dans des champs JSON imbriqués n'est pas pris en charge. Le connecteur ne mappe que les champs de niveau supérieur.
- Le connecteur ne prend pas en charge les types complexes. Les types complexes sont interprétés comme des chaînes.
- Pour extraire ou travailler avec des valeurs JSON complexes, utilisez les fonctions relatives à JSON disponibles dans Athena. Pour plus d'informations, consultez Extraction de données JSON à partir de chaînes.
- Le connecteur ne prend pas en charge l'accès aux métadonnées des messages Kafka.

Conditions

- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Point de terminaison Kafka – Chaîne de texte qui établit une connexion à une instance Kafka.

Compatibilité des clusters

Le connecteur MSK peut être utilisé avec les types de clusters suivants.

- Cluster provisionné MSK – Vous spécifiez, surveillez et adaptez manuellement la capacité du cluster.
- Cluster sans serveur MSK – Il fournit une capacité à la demande qui évolue automatiquement en fonction de l'évolution des E/S des applications.
- Kafka autonome – Connexion directe à Kafka (authentifiée ou non).

Méthodes d'authentification prises en charge

Le connecteur prend en charge les méthodes d'authentification suivantes.

- SASL/IAM
- SSL
- SASL/SCRAM
- SASL/PLAIN
- SASL/PLAINTEXT
- NO_AUTH

  Pour plus d’informations, consultez Configuration de l'authentification pour le connecteur Athena MSK.

Formats de données d'entrée pris en charge

Le connecteur prend en charge les formats de données d'entrée suivants.

- JSON
- CSV

Paramètres

Utilisez les variables d'environnement Lambda mentionnées dans cette rubrique pour configurer le connecteur MSK d'Athena.

- auth_type – Spécifie le type d'authentification du cluster. Le connecteur prend en charge les types d'authentification suivants :
  - NO_AUTH – Connectez-vous directement à Kafka sans authentification (par exemple, à un cluster Kafka déployé sur une instance EC2 qui n'utilise pas l'authentification).
  - SASL_SSL_PLAIN – Cette méthode utilise le protocole de sécurité `SASL_SSL` et le mécanisme SASL `PLAIN`.
  - SASL_PLAINTEXT_PLAIN – Cette méthode utilise le protocole de sécurité `SASL_PLAINTEXT` et le mécanisme SASL `PLAIN`.

    > **Note**
    >
    > Les types d'authentification `SASL_SSL_PLAIN` et `SASL_PLAINTEXT_PLAIN` sont pris en charge par Apache Kafka, mais pas par Amazon MSK.

  - SASL_SSL_AWS_MSK_IAM – Le contrôle d'accès IAM pour Amazon MSK vous permet de gérer à la fois l'authentification et l'autorisation pour votre cluster MSK. Les informations d' AWS identification de votre utilisateur (clé secrète et clé d'accès) sont utilisées pour se connecter au cluster. Pour plus d'informations, voir la rubrique Contrôle d'accès IAM du Guide du développeur Amazon Managed Streaming for Apache Kafka.
  - SASL_SSL_SCRAM_SHA512 – Vous pouvez utiliser ce type d'authentification pour contrôler l'accès à vos clusters Amazon MSK. Cette méthode enregistre le nom d'utilisateur et le mot de passe sur AWS Secrets Manager. Le secret doit être associé au cluster Amazon MSK. Pour plus d'informations, voir la rubrique Configuration de l'authentification SASL/SCRAM pour un cluster Amazon MSK du Guide du développeur Amazon Managed Streaming for Apache Kafka.
  - SSL – L'authentification SSL utilise des fichiers keystore et truststore pour se connecter au cluster Amazon MSK. Vous devez générer les fichiers keystore et truststore, les charger dans un compartiment Amazon S3 et fournir la référence à Amazon S3 lorsque vous déployez le connecteur. Les clés keystore, truststore et SSL sont stockées dans AWS Secrets Manager. Votre client doit fournir la clé AWS secrète lors du déploiement du connecteur. Pour plus d'informations, voir la rubrique Authentification TLS mutuelle du Guide du développeur Amazon Managed Streaming for Apache Kafka.

    Pour plus d'informations, consultez Configuration de l'authentification pour le connecteur Athena MSK.
- certificates_s3_reference – L'emplacement Amazon S3 qui contient les certificats (les fichiers keystore et truststore).

- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.
- kafka_endpoint – Les détails du point de terminaison à fournir à Kafka. Par exemple, pour un cluster Amazon MSK, vous fournissez une URL d'amorçage pour le cluster.
- secrets_manager_secret – Le nom du secret AWS dans lequel sont enregistrées les informations d'identification. Ce paramètre n'est pas obligatoire pour l'authentification IAM.
- Paramètres de déversement – Les fonctions Lambda stockent temporairement (« déversent ») les données qui ne tiennent pas en mémoire vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement. Utilisez les paramètres du tableau suivant pour spécifier l'emplacement de déversement.

Paramètre	Description
`spill_bucket`	Obligatoire. Le nom du compartiment Amazon S3 où la fonction Lambda peut déverser des données.
`spill_prefix`	Obligatoire. Le préfixe dans le compartiment de déversement où la fonction Lambda peut déverser des données.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant indique les types de données correspondants pris en charge par Kafka et Apache Arrow.

Kafka	Flèche
CHAR	VARCHAR
VARCHAR	VARCHAR
TIMESTAMP	MILLISECOND
DATE	DAY
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	FLOAT8
DOUBLE	FLOAT8

Partitions et déversements

Les rubriques Kafka sont divisées en partitions. Chaque partition est ordonnée. Chaque message dans une partition a un ID incrémentiel appelé offset. Chaque partition Kafka est ensuite divisée en plusieurs parties pour un traitement parallèle. Les données sont disponibles pour la période de rétention configurée dans les clusters Kafka.

Bonnes pratiques

Il est recommandé d'utiliser la poussée vers le bas des prédicats lorsque vous interrogez Athena, comme dans les exemples suivants.

```
SELECT *
FROM "msk_catalog_name"."glue_schema_registry_name"."glue_schema_name"
WHERE integercol = 2147483647
```

```
SELECT *
FROM "msk_catalog_name"."glue_schema_registry_name"."glue_schema_name"
```

```
WHERE timestampcol >= TIMESTAMP '2018-03-25 07:30:58.878'
```

Configuration du connecteur MSK

Pour pouvoir utiliser le connecteur, vous devez configurer votre cluster Amazon MSK, utiliser le Registre de schémas AWS Glue pour définir votre schéma et configurer l'authentification pour le connecteur.

> **Note**
>
> Si vous déployez le connecteur dans un VPC afin d'accéder à des ressources privées et si vous souhaitez également vous connecter à un service accessible au public tel que Confluent, vous devez associer le connecteur à un sous-réseau privé doté d'une passerelle NAT. Pour plus d'informations, consultez Passerelles NAT dans le Guide de l'utilisateur Amazon VPC.

Lorsque vous travaillez avec le registre des AWS Glue schémas, tenez compte des points suivants :

- Assurez-vous que le texte du champ Description du registre de schémas AWS Glue inclut la chaîne `{AthenaFederationMSK}`. Cette chaîne de marqueur est obligatoire pour les AWS Glue registres que vous utilisez avec le connecteur Amazon Athena MSK.
- Pour des performances optimales, n'utilisez que des minuscules pour vos noms de bases de données et de tables. L'utilisation d'une casse mixte oblige le connecteur à effectuer une recherche insensible à la casse, ce qui demande plus de temps de calcul.

Pour configurer votre environnement Amazon MSK et votre registre de AWS Glue schémas

1. Configurez votre environnement Amazon MSK. Pour plus d'informations et d'étapes, voir la rubrique Configuration d'Amazon MSK et Mise en route avec Amazon MSK dans le Guide du développeur Amazon Managed Streaming for Apache Kafka.
2. Téléchargez le fichier de description de la rubrique Kafka (c'est-à-dire son schéma) au format JSON dans le registre des AWS Glue schémas. Pour plus d'informations, consultez la section Intégration à AWS Glue Schema Registry dans le guide du AWS Glue développeur. Pour des exemples de schémas, voir la rubrique suivante.

Exemples de schémas pour le registre de schémas AWS Glue

Utilisez le format des exemples de cette rubrique lorsque vous téléchargez votre schéma dans le registre de schémas AWS Glue.

Exemple de schéma de type JSON

Dans l'exemple suivant, le schéma à créer dans le registre des AWS Glue `json` schémas indique la valeur pour `dataFormat` et les utilisations `datatypejson` pour`topicName`.

> **Note**
>
> La valeur de `topicName` doit utiliser la même casse que le nom de la rubrique de Kafka.

```
{
  "topicName": "datatypejson",
  "message": {
    "dataFormat": "json",
    "fields": [
      {
        "name": "intcol",
        "mapping": "intcol",
        "type": "INTEGER"
      },
      {
        "name": "varcharcol",
        "mapping": "varcharcol",
        "type": "VARCHAR"
      },
      {
        "name": "booleancol",
        "mapping": "booleancol",
        "type": "BOOLEAN"
      },
      {
        "name": "bigintcol",
        "mapping": "bigintcol",
        "type": "BIGINT"
      },
      {
        "name": "doublecol",
        "mapping": "doublecol",
```

```
        "type": "DOUBLE"
      },
      {
        "name": "smallintcol",
        "mapping": "smallintcol",
        "type": "SMALLINT"
      },
      {
        "name": "tinyintcol",
        "mapping": "tinyintcol",
        "type": "TINYINT"
      },
      {
        "name": "datecol",
        "mapping": "datecol",
        "type": "DATE",
        "formatHint": "yyyy-MM-dd"
      },
      {
        "name": "timestampcol",
        "mapping": "timestampcol",
        "type": "TIMESTAMP",
        "formatHint": "yyyy-MM-dd HH:mm:ss.SSS"
      }
    ]
  }
}
```

Exemple de schéma de type CSV

Dans l'exemple suivant, le schéma à créer dans le registre des AWS Glue `csv` schémas indique la valeur pour `dataFormat` et les utilisations `datatypecsvbulk` pour`topicName`. La valeur de `topicName` doit utiliser la même casse que le nom de la rubrique de Kafka.

```
{
  "topicName": "datatypecsvbulk",
  "message": {
    "dataFormat": "csv",
    "fields": [
      {
        "name": "intcol",
        "type": "INTEGER",
        "mapping": "0"
```

```
      },
      {
        "name": "varcharcol",
        "type": "VARCHAR",
        "mapping": "1"
      },
      {
        "name": "booleancol",
        "type": "BOOLEAN",
        "mapping": "2"
      },
      {
        "name": "bigintcol",
        "type": "BIGINT",
        "mapping": "3"
      },
      {
        "name": "doublecol",
        "type": "DOUBLE",
        "mapping": "4"
      },
      {
        "name": "smallintcol",
        "type": "SMALLINT",
        "mapping": "5"
      },
      {
        "name": "tinyintcol",
        "type": "TINYINT",
        "mapping": "6"
      },
      {
        "name": "floatcol",
        "type": "DOUBLE",
        "mapping": "7"
      }
    ]
  }
}
```

Configuration de l'authentification pour le connecteur Athena MSK

Vous pouvez utiliser diverses méthodes pour vous authentifier dans votre cluster Amazon MSK, notamment IAM, SSL, SCRAM et Kafka autonome.

Le tableau suivant présente les types d'authentification pour le connecteur ainsi que le protocole de sécurité et le mécanisme SASL pour chacun d'entre eux. Pour plus d'informations, voir la rubrique Authentification et autorisation pour les API Apache Kafka du Guide du développeur Amazon Managed Streaming for Apache Kafka.

auth_type	security.protocol	sasl.mechanism
`SASL_SSL_PLAIN`	`SASL_SSL`	`PLAIN`
`SASL_PLAINTEXT_PLAIN`	`SASL_PLAINTEXT`	`PLAIN`
`SASL_SSL_AWS_MSK_IAM`	`SASL_SSL`	`AWS_MSK_IAM`
`SASL_SSL_SCRAM_SHA512`	`SASL_SSL`	`SCRAM-SHA-512`
`SSL`	`SSL`	N/A

> **Note**
>
> Les types d'authentification `SASL_SSL_PLAIN` et `SASL_PLAINTEXT_PLAIN` sont pris en charge par Apache Kafka, mais pas par Amazon MSK.

SASL/IAM

Si le cluster utilise l'authentification IAM, vous devez configurer la politique IAM pour l'utilisateur lorsque vous configurez le cluster. Pour plus d'informations, voir la rubrique Contrôle d'accès IAM du Guide du développeur Amazon Managed Streaming for Apache Kafka.

Pour utiliser ce type d'authentification, définissez la variable d'environnement Lambda `auth_type` pour le connecteur sur `SASL_SSL_AWS_MSK_IAM`.

SSL

Si le cluster est authentifié SSL, vous devez générer les fichiers truststore (magasin d'approbations) et keystore (magasin de clés) et les télécharger dans le compartiment Amazon S3. Vous devez fournir cette référence Amazon S3 lorsque vous déployez le connecteur. Les clés keystore, truststore et SSL sont stockées dans AWS Secrets Manager. Vous fournissez la clé AWS secrète lorsque vous déployez le connecteur.

Pour plus d'informations sur la création d'un secret dans Secrets Manager, voir la rubrique Créer un secret AWS Secrets Manager.

Pour utiliser ce type d'authentification, définissez les variables d'environnement comme indiqué dans le tableau suivant.

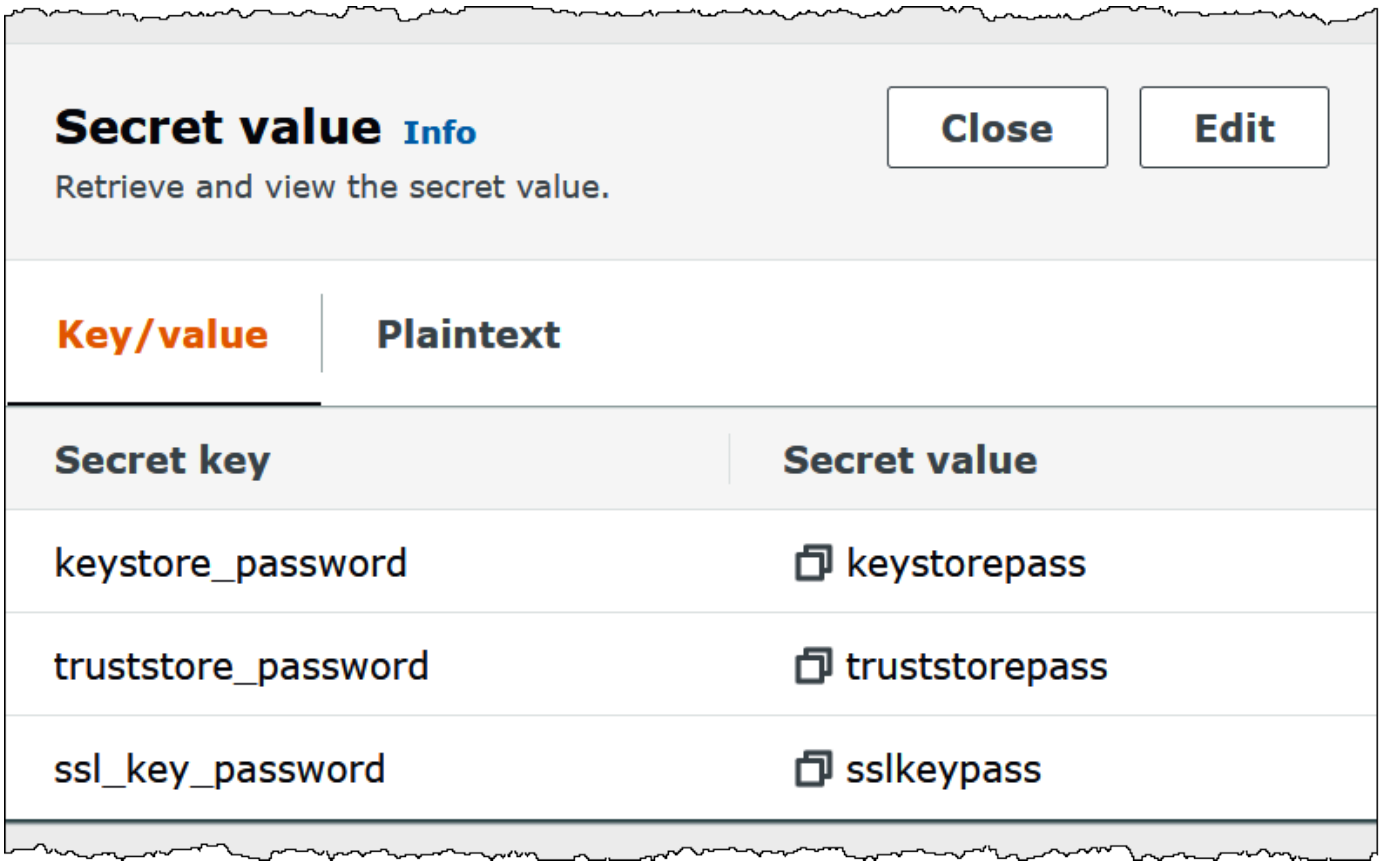
Paramètre	Valeur
`auth_type`	`SSL`
`certificates_s3_reference`	L'emplacement Amazon S3 qui contient les certificats.
`secrets_manager_secret`	Le nom de votre clé AWS secrète.

Après avoir créé un secret dans Secrets Manager, vous pouvez le consulter dans la console Secrets Manager.

Visualisation de votre secret dans Secrets Manager

1. Ouvrez la console Secrets Manager en suivant le lien https://console.aws.amazon.com/secretsmanager/.
2. Dans le volet de navigation, sélectionnez Secrets.
3. Sur la page Secrets, choisissez le lien vers votre secret.
4. Sur la page de détails de votre secret, sélectionnez Retrieve secret value (Récupérer la valeur du secret).

   L'image suivante montre un exemple de secret avec trois paires clé/valeur : `keystore_password`, `truststore_password` et `ssl_key_password`.



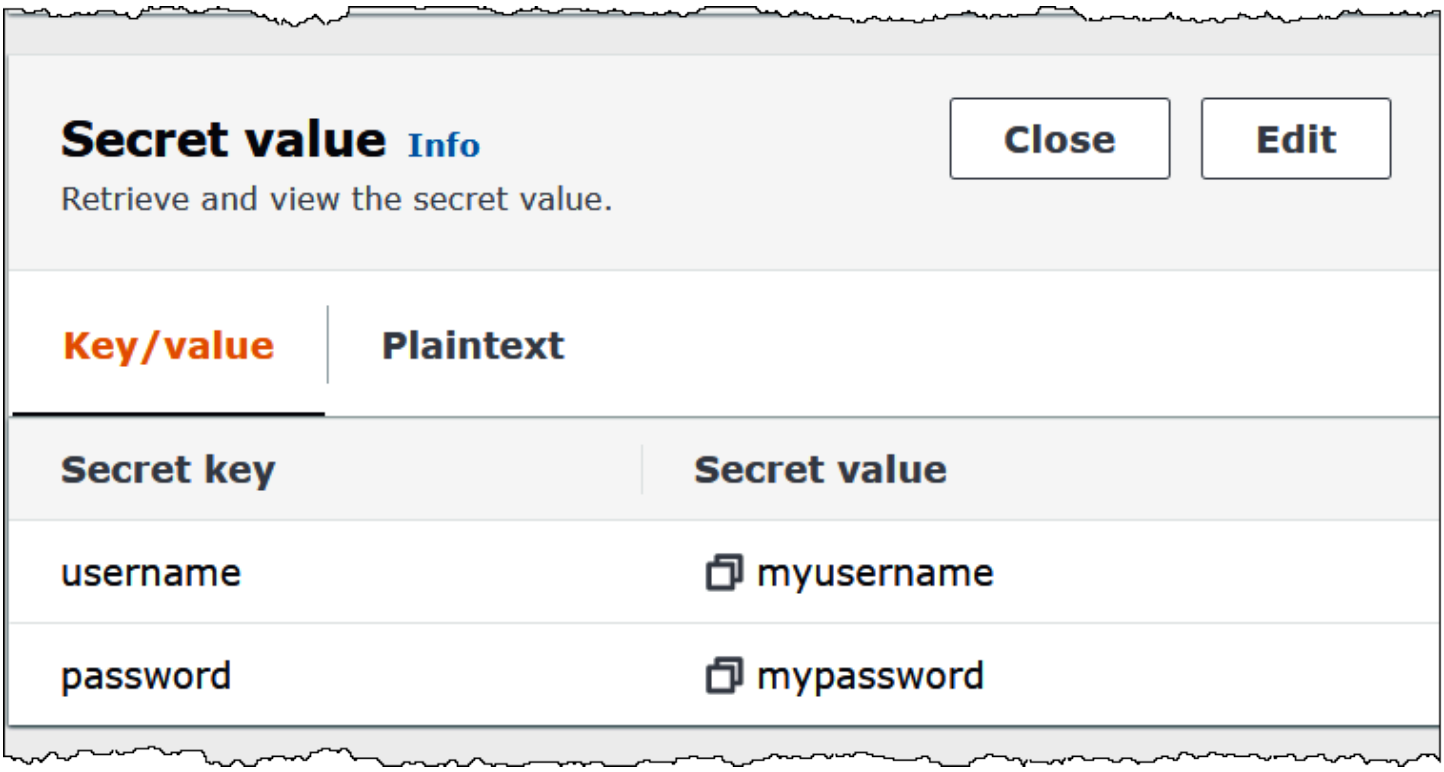
SASL/SCRAM

Si votre cluster utilise l'authentification SCRAM, fournissez la clé Secrets Manager qui est associée au cluster lorsque vous déployez le connecteur. Les informations d'identification AWS de l'utilisateur (clé secrète et clé d'accès) sont utilisées pour s'authentifier au cluster.

Définissez les variables d'environnement comme indiqué dans le tableau suivant.

Paramètre	Valeur
`auth_type`	`SASL_SSL_SCRAM_SHA512`
`secrets_manager_secret`	Le nom de votre clé AWS secrète.

L'image suivante montre un exemple de secret dans la console Secrets Manager avec deux paires clé/valeur : une pour `username` et une pour `password`.



Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour MySQL

Le connecteur Lambda MySQL Amazon Athena permet à Amazon Athena d'accéder à vos bases de données MySQL.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.
- Avant d'utiliser ce connecteur, configurez un VPC et un groupe de sécurité. Pour plus d'informations, consultez Création d'un VPC pour un connecteur de source de données.

Limites

- Les opérations DDL d’écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Comme Athena convertit les requêtes en minuscules, les noms des tables MySQL doivent être en minuscules. Par exemple, les requêtes Athena par rapport à une table nommée `myTable` échoue.

Conditions

Les termes suivants se rapportent au connecteur MySQ.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d’enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d’environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur MySQL.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
mysql://${jdbc_connection_string}
```

> **Note**
>
> Si vous recevez le message d'erreur java.sql.SQLException: Zero date value prohibited (java.sql.SQL.SqlException : valeur de date nulle interdite) lorsque vous effectuez une requête `SELECT` sur une table MySQL, ajoutez le paramètre suivant à votre chaîne de connexion :
>
> ```
> zeroDateTimeBehavior=convertToNull
> ```
>
> Pour plus d'informations, consultez l'erreur « Valeur de date zéro interdite » lors de la tentative de sélection dans la table MySQL sur .com. GitHub

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`MySqlMuxCompositeHandler`
Gestionnaire de métadonnées	`MySqlMuxMetadataHandler`

Handler (Gestionnaire)	Classe
Gestionnaire d'enregis trements	`MySqlMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
$*catalog*_connecti on_string	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `mymysqlcatalog` , le nom de la variable d'environnement est alors `mymysqlcatalog_connection_s tring` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *AWS_LAMBD A_FUNCTION_NAME* `}`.

Les exemples de propriétés suivants concernent une fonction Lambda MySql MUX qui prend en charge deux instances de base de données `mysql1` : (par défaut) et. `mysql2`

Propriété	Valeur
`default`	`mysql://jdbc:mysql://mysql2 .host:3333/default? user=samp le2&password=sample2`
`mysql_catalog1_connection_string`	`mysql://jdbc:mysql://mysql1 .host:3306/default?${Test/RDS/ MySql1}`
`mysql_catalog2_connection_string`	`mysql://jdbc:mysql://mysql2 .host:3333/default? user=samp le2&password=sample2`

Fourniture des informations d’identification

Pour fournir un nom d’utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

  Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n’utilise pas Amazon RDS, stockez les informations d’identification au format JSON au format suivant :

  ```
  {"username": "${username}", "password": "${password}"}
  ```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${Test/RDS/MySql1}`.

```
mysql://jdbc:mysql://mysql1.host:3306/default?...&${Test/RDS/MySql1}&...
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
mysql://jdbc:mysql://mysql1host:3306/default?...&user=sample2&password=sample2&...
```

Actuellement, le connecteur MySQL reconnaît les propriétés JDBC `user` et `password`.

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance MySQL.

Type de gestionnaire	Classe
Gestionnaire de composites	`MySqlCompositeHandler`
Gestionnaire de métadonnées	`MySqlMetadataHandler`
Gestionnaire d'enregis trements	`MySqlRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une instance MySQL unique prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`mysql://mysql1.host:3306/default?secret=Test/RDS/` MySql1

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant affiche les types de données correspondants pour JDBC et Arrow.

JDBC	Flèche
Booléen	Bit
Entier	Tiny
Court	Smallint
Entier	Int
Long	Bigint

JDBC	Flèche
float	Float4
Double	Float8
Date	DateDay
Horodatage	DateMilli
Chaîne	Varchar
Octets	Varbinary
BigDecimal	Décimal
ARRAY	Liste

Partitions et déversements

Les partitions sont utilisées pour déterminer comment générer des divisions pour le connecteur. Athena construit une colonne synthétique de type `varchar` qui représente le schéma de partitionnement de la table afin d'aider le connecteur à générer des divisions. Le connecteur ne modifie pas la définition réelle de la table.

Performance

MySQL prend en charge les partitions natives. Le connecteur Athena MySQL peut récupérer les données de ces partitions en parallèle. Si vous souhaitez interroger des jeux de données très volumineux avec une distribution de partition uniforme, le partitionnement natif est fortement recommandé.

Le connecteur Athena MySQL effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Les clauses `LIMIT`, les prédicats simples et les expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution des requêtes.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena MySQL peut combiner ces expressions et les pousser directement vers MySQL pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena MySQL suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%')
LIMIT 10;
```

Pour un article sur l'utilisation de la poussée vers le bas de prédicats pour améliorer les performances des requêtes fédérées, y compris MySQL, consultez Améliorer les requêtes fédérées avec la poussée vers le bas de prédicats dans Amazon Athena sur le blog AWS Big Data.

Requêtes directes

Le connecteur MySQL prend en charge les requêtes passthrough. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes passthrough avec MySQL, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans MySQL. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les dernières informations sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur MySQL sur GitHub .com.

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Neptune

Amazon Neptune est un service de base de données orientée graphe entièrement géré et fiable, qui facilite la création et l'exécution d'applications fonctionnant avec des jeux de données hautement connectés. Neptune conçu spécialement, haute performance, le moteur de base de données graphique stocke des milliards de relations de manière optimale et des requêtes sur les graphiques avec une latence de seulement quelques millisecondes. Pour plus d'informations, consultez Neptune Guide de l'utilisateur.

Le connecteur Amazon Athena Neptune permet à Athena de communiquer avec votre instance de base de données orientée graphe Neptune, rendant vos données de graphe Neptune accessibles par des requêtes SQL.

Si Lake Formation est activé sur votre compte, le rôle IAM de votre connecteur Lambda fédéré Athena que vous avez déployé dans AWS Serverless Application Repository le doit avoir un accès en lecture dans Lake Formation au. AWS Glue Data Catalog

Prérequis

L'utilisation du connecteur Neptune nécessite les trois étapes suivantes.

- Configuration d'un cluster Neptune
- Configuration d'un AWS Glue Data Catalog
- Déploiement du connecteur sur votre Compte AWS. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données. Pour plus d'informations spécifiques au déploiement du connecteur Neptune, consultez Déployer le connecteur Amazon Athena Neptune sur .com. GitHub

Limites

Actuellement, le connecteur Neptune présente les limites suivantes.

- La projection de colonnes, y compris la clé primaire (ID), n'est pas prise en charge.

Configuration d'un cluster Neptune

Si vous ne disposez pas d'un cluster Amazon Neptune et d'un jeu de données de graphes de propriétés que vous souhaitez utiliser, vous devez en configurer un.

Assurez-vous de disposer d'une passerelle Internet et d'une passerelle NAT dans le VPC qui héberge votre cluster Neptune. Les sous-réseaux privés utilisés par la fonction Lambda du connecteur Neptune doivent avoir une route vers Internet via cette passerelle NAT. La fonction Lambda du connecteur Neptune utilise la passerelle NAT pour communiquer avec. AWS Glue

Pour obtenir des instructions sur la configuration d'un nouveau cluster Neptune et son chargement avec un exemple de jeu de données, consultez Sample Neptune Cluster Setup sur .com. GitHub

Configuration d'un AWS Glue Data Catalog

Contrairement aux magasins de données relatives traditionnels, les nœuds et les périphéries de base de données de graphe Neptune n'utilisent pas de schéma défini. Chaque entrée peut comporter des

champs et des types de données différents. Toutefois, étant donné que le connecteur Neptune extrait les métadonnées du AWS Glue Data Catalog, vous devez créer une AWS Glue base de données contenant des tables avec le schéma requis. Après avoir créé la base de données et les tables AWS Glue , le connecteur peut fournir la liste des tables disponibles pour effectuer des requêtes auprès d'Athena.

Activation de la mise en correspondance non sensible à la casse des colonnes

Pour résoudre les noms de colonnes de votre table Neptune avec le boitier correct, même si les noms de colonnes sont tous en minuscules AWS Glue, vous pouvez configurer le connecteur Neptune pour une correspondance insensible aux majuscules et minuscules.

Pour activer cette fonctionnalité, définissez la variable d'environnement de la fonction Lambda du connecteur Neptune de `enable_caseinsensitivematch` à `true`.

Spécification du paramètre de table AWS Glue glabel pour les noms de table en majuscules

Étant donné que seuls AWS Glue les noms de table en minuscules sont pris en charge, il est important de spécifier le paramètre de `glabel` AWS Glue table lorsque vous créez une AWS Glue table pour Neptune et que le nom de votre table Neptune inclut le boîtier.

Dans la définition de votre AWS Glue table, incluez le `glabel` paramètre et attribuez à sa valeur le nom de votre table avec son boîtier d'origine. Cela garantit que le boîtier correct est préservé lors de AWS Glue l'interaction avec votre table Neptune. L'exemple suivant définit la valeur de `glabel` sur le nom de la table `Airport`.

```
glabel = Airport
```

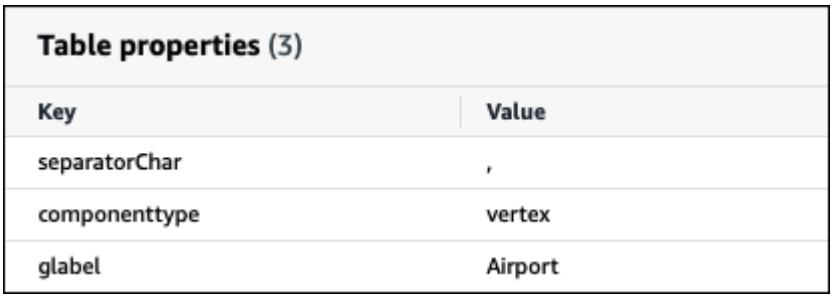


Table properties (3)

Key	Value
separatorChar	,
componenttype	vertex
glabel	Airport

Pour plus d'informations sur la configuration d'un AWS Glue Data Catalog pour qu'il fonctionne avec Neptune, voir Configurer le AWS Glue catalogue sur GitHub .com.

Performance

Le connecteur Athena Neptune effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Cependant, les prédicats utilisant la clé primaire entraînent l'échec de la requête. Les clauses `LIMIT` réduisent la quantité de données analysées, mais si vous ne fournissez pas de prédicat, vous devez vous attendre à ce que les requêtes `SELECT` avec une clause `LIMIT` analysent au moins 16 Mo de données. Le connecteur Neptune résiste à la limitation due à la simultanéité.

Requêtes passthrough

Le connecteur Neptune prend en charge les requêtes directes. Vous pouvez utiliser cette fonctionnalité pour exécuter des requêtes Gremlin sur des graphes de propriétés et pour exécuter des requêtes SPARQL sur des données RDF.

Pour créer des requêtes directes avec Neptune, utilisez la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'database_name',
            COLLECTION => 'collection_name',
            QUERY => 'query_string'
        ))
```

L'exemple suivant donne un exemple de filtre de requête Neptune passthrough pour les aéroports utilisant le code. ATL Les guillemets simples doublés servent à s'échapper.

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'graph-database',
            COLLECTION => 'airport',
            QUERY => 'g.V().has(''airport'', ''code'', ''ATL'').valueMap()'
        ))
```

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena OpenSearch

OpenSearch Service

Le OpenSearch connecteur Amazon Athena permet à Amazon Athena de communiquer avec OpenSearch vos instances afin que vous puissiez utiliser le SQL pour interroger vos données. OpenSearch

> **Note**
>
> En raison d'un problème connu, le OpenSearch connecteur ne peut pas être utilisé avec un VPC.

Si Lake Formation est activé sur votre compte, le rôle IAM de votre connecteur Lambda fédéré Athena que vous avez déployé dans AWS Serverless Application Repository le doit avoir un accès en lecture dans Lake Formation au. AWS Glue Data Catalog

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Conditions

Les termes suivants concernent le OpenSearch connecteur.

- Domaine : nom que ce connecteur associe au point de terminaison de votre OpenSearch instance. Le domaine est également utilisé comme nom de base de données. Pour les OpenSearch instances définies au sein d'Amazon OpenSearch Service, le domaine est détectable automatiquement. Pour les autres cas, vous devez fournir un mappage entre le nom de domaine et le point de terminaison.
- Index — Table de base de données définie dans votre OpenSearch instance.
- Mapping (Mappage) – Si un index est une table de base de données, le mappage est son schéma (c'est-à-dire les définitions de ses champs et attributs).

  Ce connecteur prend en charge à la fois la récupération des métadonnées depuis l' OpenSearch instance et depuis le AWS Glue Data Catalog. Si le connecteur trouve une AWS Glue base de données et une table qui correspondent à vos noms de OpenSearch domaine et d'index, il tente

de les utiliser pour définir le schéma. Nous vous recommandons de créer votre AWS Glue table de manière à ce qu'elle soit un sur-ensemble de tous les champs de votre OpenSearch index.

- Document – Un enregistrement au sein d'une table de base de données.
- Flux de données – Données temporelles composées de plusieurs index de support. Pour plus d'informations, consultez les sections Flux de données dans la OpenSearch documentation et Getting started with data streams dans le Amazon OpenSearch Service Developer Guide.

> **Note**
>
> Les index de flux de données étant créés et gérés en interne par OpenSearch, le connecteur choisit le mappage de schéma à partir du premier index disponible. Pour cette raison, nous vous recommandons vivement de configurer une AWS Glue table comme source de métadonnées supplémentaire. Pour plus d'informations, consultez Configuration de bases de données et de tables dans AWS Glue.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le OpenSearch connecteur.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.
- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.
- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.
- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.

- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.
- disable_glue — (Facultatif) S'il est présent et défini sur true, le connecteur ne tente pas de récupérer des métadonnées supplémentaires à partir de. AWS Glue
- query_timeout_cluster – Le délai d'expiration, en secondes, pour les requêtes d'intégrité du cluster utilisées dans la génération d'examens parallèles.
- query_timeout_search – Le délai d'expiration, en secondes, des requêtes de recherche utilisées pour récupérer des documents à partir d'un index.
- auto_discover_endpoint – Valeur booléenne. L'argument par défaut est `true`. Lorsque vous utilisez le OpenSearch service Amazon et que vous définissez ce paramètre sur true, le connecteur peut découvrir automatiquement vos domaines et points de terminaison en appelant les opérations d'API de description ou de liste appropriées sur le OpenSearch service. Pour tout autre type d' OpenSearch instance (par exemple, auto-hébergée), vous devez spécifier les points de terminaison de domaine associés dans la `domain_mapping` variable. Si`auto_discover_endpoint=true`, le connecteur utilise des AWS informations d'identification pour s'authentifier auprès du OpenSearch service. Dans le cas contraire, le connecteur récupère les informations d'identification du nom d'utilisateur et du mot de passe AWS Secrets Manager par le biais de la `domain_mapping` variable.
- domain_mapping – Utilisé uniquement lorsque `auto_discover_endpoint` est défini sur false et définit le mappage entre les noms de domaine et leurs points de terminaison associés. La `domain_mapping` variable peut prendre en charge plusieurs OpenSearch points de terminaison au format suivant :

  ```
  domain1=endpoint1,domain2=endpoint2,domain3=endpoint3,...
  ```

  Aux fins de l'authentification auprès d'un OpenSearch point de terminaison, le connecteur prend en charge les chaînes de substitution injectées en utilisant le format dont le nom `${SecretName}:` d'utilisateur et le mot de passe ont été extraits AWS Secrets Manager. Le signe deux-points (:) à la fin de l'expression sert de séparateur par rapport au reste du point de terminaison.

> **Important**
>
> Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

L'exemple suivant utilise le secret `opensearch-creds`.

```
movies=https://${opensearch-creds}:search-movies-ne...qu.us-east-1.es.amazonaws.com
```

Au moment de l'exécution, `${opensearch-creds}` est affiché sous la forme du nom d'utilisateur et du mot de passe, comme dans l'exemple suivant.

```
movies=https://myusername@mypassword:search-movies-ne...qu.us-east-1.es.amazonaws.com
```

Dans le paramètre `domain_mapping`, chaque paire domaine-point de terminaison peut utiliser un secret différent. Le secret lui-même doit être spécifié au format *`nom_utilisateur`*`@`*`mot de passe`*. Bien que le mot de passe puisse contenir des signes `@` intégrés, le premier `@` sert de séparateur de*`nom_utilisateur`*.

Il est également important de noter que la virgule (,) et le signe égal (=) sont utilisés par ce connecteur comme séparateurs pour les paires domaine-point de terminaison. Pour cette raison, vous ne devez les utiliser nulle part dans le secret stocké.

Configuration de bases de données et de tables dans AWS Glue

Le connecteur obtient des informations de métadonnées à l'aide de AWS Glue ou OpenSearch. Vous pouvez configurer une AWS Glue table comme source de définition de métadonnées supplémentaire. Pour activer cette fonctionnalité, définissez une AWS Glue base de données et une table correspondant au domaine et à l'index de la source que vous complétez. Le connecteur peut également tirer parti des définitions de métadonnées stockées dans l' OpenSearch instance en récupérant le mappage pour l'index spécifié.

Définition des métadonnées pour les tableaux dans OpenSearch

OpenSearch ne possède pas de type de données de tableau dédié. Chaque champ peut contenir zéro ou plusieurs valeurs, à condition qu'elles soient du même type de données. Si vous souhaitez l'utiliser OpenSearch comme source de définition de métadonnées, vous devez définir une `_meta` propriété pour tous les index utilisés avec Athena pour les champs qui doivent être considérés comme une liste ou un tableau. Si vous ne parvenez pas à effectuer cette étape, les requêtes renvoient uniquement le premier élément du champ de liste. Lorsque vous spécifiez la propriété `_meta`, les noms de champs doivent être entièrement qualifiés pour les structures JSON imbriquées (par exemple, `address.street`, où `street` est un champ imbriqué dans une structure `address`).

L'exemple suivant définit les listes `actor` et `genre` dans la table `movies`.

```
PUT movies/_mapping
{
  "_meta": {
    "actor": "list",
    "genre": "list"
  }
}
```

Types de données

Le OpenSearch connecteur peut extraire les définitions de métadonnées de l'instance AWS Glue ou de l' OpenSearch instance. Le connecteur utilise le mappage du tableau suivant pour convertir les définitions en types de données Apache Arrow, y compris les points notés dans la section suivante.

OpenSearch	Apache	AWS Glue
texte, mot clé, binaire	VARCHAR	chaîne
long	BIGINT	bigint
scaled_float	BIGINT	SCALED_FLOAT(...)
entier	INT	int
short	SMALLINT	smallint
octet	TINYINT	tinyint

OpenSearch	Apache	AWS Glue
double	FLOAT8	double
float, half_float	FLOAT4	float
boolean	BIT	boolean
date, date_nanos	DATEMILLI	timestamp
Structure JSON	STRUCT	STRUCT
_meta (pour plus d'informations, consultez la section Définition des métadonnées pour les tableaux dans OpenSearch.)	LIST	ARRAY

Remarques sur les types de données

- Actuellement, le connecteur ne prend en charge que les AWS Glue types de données OpenSearch et répertoriés dans le tableau précédent.
- `Unscaled_float` est un nombre à virgule flottante mis à l'échelle par un facteur de mise à l'échelle double fixe et représenté en tant que `BIGINT` dans Apache Arrow. Par exemple, 0,756 avec un facteur d'échelle de 100 est arrondi à 76.
- *Pour définir un scaled_float in AWS Glue, vous devez sélectionner le type de array colonne et déclarer le champ au format SCALED_FLOAT (scaling_factor).*

  Les exemples suivants sont valides :

  ```
  SCALED_FLOAT(10.51)
  SCALED_FLOAT(100)
  SCALED_FLOAT(100.0)
  ```

  Les exemples suivants ne sont pas valides :

  ```
  SCALED_FLOAT(10.)
  ```

```
SCALED_FLOAT(.5)
```

- Lors de la conversion de `date_nanos` vers `DATEMILLI`, les nanosecondes sont arrondies à la milliseconde la plus proche. Valeurs valides pour `date` et `date_nanos` incluent, sans s'y limiter, les formats suivants :

```
"2020-05-18T10:15:30.123456789"
"2020-05-15T06:50:01.123Z"
"2020-05-15T06:49:30.123-05:00"
1589525370001 (epoch milliseconds)
```

- An OpenSearch `binary` est une représentation sous forme de chaîne d'une valeur binaire codée en utilisant `Base64` et convertie en `VARCHAR` a.

Exécution de requêtes SQL

Vous trouverez ci-dessous des exemples de requêtes DDL que vous pouvez utiliser avec ce connecteur. Dans les exemples,*`function_name`* correspond au nom de votre fonction Lambda,*`domaine`* est le nom du domaine à interroger et *`index`* est le nom de votre index.

```
SHOW DATABASES in `lambda:function_name`
```

```
SHOW TABLES in `lambda:function_name`.domain
```

```
DESCRIBE `lambda:function_name`.domain.index
```

Performance

Le OpenSearch connecteur Athena prend en charge les scans parallèles basés sur des partitions. Le connecteur utilise les informations relatives à l'état du cluster extraites de l' OpenSearch instance pour générer plusieurs demandes de recherche de documents. Les demandes sont réparties pour chaque partition et exécutées simultanément.

Le connecteur transfère également des prédicats dans le cadre de ses requêtes de recherche de documents. L'exemple de requête et de prédicat suivant montre comment le connecteur utilise le prédicat poussé vers le bas.

Interrogation

```
SELECT * FROM "lambda:elasticsearch".movies.movies
WHERE year >= 1955 AND year <= 1962 OR year = 1996
```

Prédicat

```
(_exists_:year) AND year:([1955 TO 1962] OR 1996)
```

Requêtes directes

Le OpenSearch connecteur prend en charge les requêtes directes et utilise le langage Query DSL. Pour plus d'informations sur les requêtes avec Query DSL, consultez Query DSL dans la documentation Elasticsearch ou Query DSL dans la documentation. OpenSearch

Pour utiliser des requêtes directes avec le OpenSearch connecteur, utilisez la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            schema => 'schema_name',
            index => 'index_name',
            query => "{query_string}"
        ))
```

L' OpenSearch exemple de requête directe suivant filtre les employés dont le statut professionnel est actif dans l'`employee`index du `default` schéma.

```
SELECT * FROM TABLE(
        system.query(
            schema => 'default',
            index => 'employee',
            query => "{ ''bool'':{''filter'':{''term'':{''status'': ''active''}}}}"
        ))
```

Ressources supplémentaires

- Pour un article sur l'utilisation du OpenSearch connecteur Amazon Athena pour interroger des données dans Amazon OpenSearch Service et Amazon S3 en une seule requête, voir Interroger des données dans Amazon OpenSearch Service à l'aide de SQL depuis Amazon Athena dans AWS le blog Big Data.
- Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Oracle

Le connecteur Amazon Athena pour Oracle permet à Amazon Athena d'exécuter des requêtes SQL sur des données stockées dans Oracle exécutées sur site ou sur Amazon EC2 ou Amazon RDS. Vous pouvez également utiliser le connecteur pour interroger des données sur Oracle Exadata.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .

Conditions

Les termes suivants se rapportent au connecteur Oracle.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.

- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Oracle.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
oracle://${jdbc_connection_string}
```

> **Note**
>
> Si votre mot de passe contient des caractères spéciaux (par exemple,`some.password`), mettez-le entre guillemets lorsque vous le transmettez à la chaîne de connexion (par exemple,`"some.password"`). Si vous ne le faites pas, une erreur d'URL Oracle spécifiée invalide peut se produire.

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`OracleMuxCompositeHandler`
Gestionnaire de métadonnées	`OracleMuxMetadataHandler`
Gestionnaire d'enregis trements	`OracleMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
$*catalog*_connecti on_string	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `myoraclecatalog` , le nom de la variable d'environnement est alors `myoraclecatalog_connection_ string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *AWS_LAMBD A_FUNCTION_NAME* `}`.

Les exemples de propriétés suivants concernent une fonction Oracle MUX Lambda qui prend en charge deux instances de base de données :`oracle1` (par défaut) et `oracle2`.

Propriété	Valeur
`default`	`oracle://jdbc:oracle:thin:$ {Test/RDS/Oracle1}@//oracle 1.hostname:port/servicename`

Propriété	Valeur
`oracle_catalog1_connection_string`	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle1}@//oracle1.hostname:port/servicename`
`oracle_catalog2_connection_string`	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle2}@//oracle2.hostname:port/servicename`

Fourniture des informations d’identification

Pour fournir un nom d’utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

  Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS

Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

> **Note**
>
> Si votre mot de passe contient des caractères spéciaux (par exemple,`some.password`), mettez-le entre guillemets lorsque vous l'enregistrez dans Secrets Manager (par exemple,`"some.password"`). Si vous ne le faites pas, une erreur d'URL Oracle spécifiée invalide peut se produire.

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${Test/RDS/Oracle}`.

```
oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@//hostname:port/servicename
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
oracle://jdbc:oracle:thin:username/password@//hostname:port/servicename
```

Actuellement, le connecteur Oracle reconnaît les propriétés JDBC `UID` et `PWD`.

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance Oracle.

Type de gestionnaire	Classe
Gestionnaire de composites	`OracleCompositeHandler`
Gestionnaire de métadonnées	`OracleMetadataHandler`

Type de gestionnaire	Classe
Gestionnaire d’enregistrements	`OracleRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.
`IsFIPSEnabled`	Facultatif. Réglé sur `true` lorsque le mode FIPS est activé. L’argument par défaut est `false`.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

Le connecteur prend en charge les connexions basées sur SSL aux instances Amazon RDS. La prise en charge est limitée au protocole TLS (Transport Layer Security) et à l'authentification du serveur par le client. L’authentification mutuelle n’est pas prise en charge dans Amazon RDS. La deuxième ligne du tableau ci-dessous indique la syntaxe pour utiliser le protocole SSL.

L’exemple de propriété suivant concerne une instance Oracle unique prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@//hostname:port/servicename`
	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS) (HOST=<HOST_NAME>)(PORT=))(CONNECT_DATA=(SID=))(SECURITY=(SSL_SERVER_CERT_DN=)))`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant indique les types de données pour JDBC, Oracle et Arrow.

JDBC	Oracle	Flèche
Booléen	boolean	Bit
Entier	N/A	Tiny
Court	smallint	Smallint
Entier	entier	Int
Long	bigint	Bigint
float	float4	Float4
Double	float8	Float8
Date	date	DateDay

JDBC	Oracle	Flèche
Horodatage	timestamp	DateMilli
Chaîne	text	Varchar
Octets	octets	Varbinary
BigDecimal	numeric(p,s)	Décimal
ARRAY	N/A (voir la remarque)	Liste

Partitions et déversements

Les partitions sont utilisées pour déterminer comment générer des divisions pour le connecteur. Athena construit une colonne synthétique de type `varchar` qui représente le schéma de partitionnement de la table afin d'aider le connecteur à générer des divisions. Le connecteur ne modifie pas la définition réelle de la table.

Performance

Oracle prend en charge les partitions natives. Le connecteur Athena Oracle peut récupérer les données de ces partitions en parallèle. Si vous souhaitez interroger des jeux de données très volumineux avec une distribution de partition uniforme, le partitionnement natif est fortement recommandé. La sélection d'un sous-ensemble de colonnes accélère considérablement l'exécution des requêtes et réduit le nombre de données analysées. Le connecteur Oracle résiste à la limitation due à la simultanéité. Cependant, les temps d'exécution des requêtes ont tendance à être longs.

Le connecteur Athena Oracle effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Des prédicats simples et des expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution de la requête.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena Oracle peut combiner ces expressions et les pousser directement vers Oracle pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena Oracle suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%');
```

Requêtes directes

Le connecteur Oracle prend en charge les requêtes directes. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Oracle, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Oracle. La requête sélectionne toutes les colonnes de la `customer` table.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les dernières informations sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur Oracle sur GitHub .com.

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour PostgreSQL

Le connecteur PostgreSQL Amazon Athena permet à Athena d’accéder à vos bases de données PostgreSQL.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l’aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d’informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d’écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .

Conditions

Les termes suivants se rapportent au connecteur PostgreSQL.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.

- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur PostgreSQL.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
postgres://${jdbc_connection_string}
```

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`PostGreSqlMuxCompositeHandler`
Gestionnaire de métadonnées	`PostGreSqlMuxMetadataHandler`
Gestionnaire d'enregis trements	`PostGreSqlMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
`$`*`catalog`*`_connecti on_string`	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `mypostgrescatalog` , le nom de la variable d'environnement est alors `mypostgrescatalog_ connection_string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *`AWS_LAMBD A_FUNCTION_NAME`* `}`.

Les exemples de propriétés suivants concernent une fonction Lambda PostGreSql MUX qui prend en charge deux instances de base de données `postgres1` : (par défaut) et. `postgres2`

Propriété	Valeur
`default`	`postgres://jdbc:postgresql://postgres1.host:5 432/default?${Test/RDS/PostGres1}`
`postgres_ catalog1_`	`postgres://jdbc:postgresql://postgres1.host:5 432/default?${Test/RDS/PostGres1}`

Propriété	Valeur
`connection_string`	
`postgres_catalog2_connection_string`	`postgres://jdbc:postgresql://postgres2.host:5432/default?user=sample&password=sample`

Fourniture des informations d’identification

Pour fournir un nom d’utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

  Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS

Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${Test/RDS/PostGres1}`.

```
postgres://jdbc:postgresql://postgres1.host:5432/default?...&${Test/RDS/PostGres1}&...
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
postgres://jdbc:postgresql://postgres1.host:5432/
default?...&user=sample2&password=sample2&...
```

Actuellement, le connecteur PostgreSQL reconnaît les propriétés JDBC `user` et `password`.

Activation de SSL

Pour prendre en charge le protocole SSL dans votre connexion PostgreSQL, ajoutez ce qui suit à votre chaîne de connexion :

```
&sslmode=verify-ca&sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

Exemple

L'exemple de chaîne de connexion suivant n'utilise pas le protocole SSL.

```
postgres://jdbc:postgresql://example-asdf-aurora-postgres-endpoint:5432/asdf?
user=someuser&password=somepassword
```

Pour activer le protocole SSL, modifiez la chaîne comme suit.

```
postgres://jdbc:postgresql://example-asdf-aurora-postgres-
endpoint:5432/asdf?user=someuser&password=somepassword&sslmode=verify-
ca&sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d’enregistrements suivants pour vous connecter à une seule instance PostgreSQL.

Type de gestionnaire	Classe
Gestionnaire de composites	`PostGreSqlCompositeHandler`
Gestionnaire de métadonnées	`PostGreSqlMetadataHandler`
Gestionnaire d’enregistrements	`PostGreSqlRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L’exemple de propriété suivant concerne une instance PostgreSQL unique prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`postgres://jdbc:postgresql://postgres1.host:5432/default?secret=${Test/RDS/PostgreSQL1}`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant indique les types de données correspondants pour JDBC, PostGre SQL et Arrow.

JDBC	PostGreSQL	Flèche
Booléen	Booléen	Bit
Entier	N/A	Tiny
Court	smallint	Smallint
Entier	entier	Int
Long	bigint	Bigint
float	float4	Float4
Double	float8	Float8
Date	date	DateDay

JDBC	PostGreSQL	Flèche
Horodatage	timestamp	DateMilli
Chaîne	text	Varchar
Octets	octets	Varbinary
BigDecimal	numeric(p,s)	Décimal
ARRAY	N/A (voir la remarque)	Liste

> **Note**
>
> Le type ARRAY est pris en charge pour le connecteur PostgreSQL avec les contraintes suivantes : les tableaux multidimensionnels (`<data_type>[][]` ou tableaux imbriqués) ne sont pas pris en charge. Les colonnes avec des types de données ARRAY non pris en charge sont converties en un tableau d'éléments de chaîne (`array<varchar>`).

Partitions et déversements

Les partitions sont utilisées pour déterminer comment générer des divisions pour le connecteur. Athena construit une colonne synthétique de type `varchar` qui représente le schéma de partitionnement de la table afin d'aider le connecteur à générer des divisions. Le connecteur ne modifie pas la définition réelle de la table.

Performance

PostgreSQL prend en charge les partitions natives. Le connecteur Athena PostgreSQL peut récupérer les données de ces partitions en parallèle. Si vous souhaitez interroger des jeux de données très volumineux avec une distribution de partition uniforme, le partitionnement natif est fortement recommandé.

Le connecteur Athena PostgreSQL effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Les clauses `LIMIT`, les prédicats simples et les expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution des requêtes. Cependant, la sélection d'un sous-ensemble de colonnes entraîne parfois un délai d'exécution plus long des requêtes.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena PostgreSQL peut combiner ces expressions et les pousser directement vers PistgreSQL pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena PostgreSQL suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%')
LIMIT 10;
```

Requêtes directes

Le connecteur PostgreSQL prend en charge les requêtes passthrough. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes passthrough avec PostgreSQL, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans PostgreSQL. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Ressources supplémentaires

Pour obtenir les dernières informations sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur PostgreSQL sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Redis

Le connecteur Amazon Athena Redis permet à Amazon Athena de communiquer avec vos instances Redis afin que vous puissiez requêter vos données Redis avec SQL. Vous pouvez utiliser le AWS Glue Data Catalog pour mapper vos paires clé-valeur Redis dans des tables virtuelles.

Contrairement aux magasins de données relatives traditionnels, Redis n'utilise pas le concept de table ou de colonne. Redis propose plutôt des modèles d'accès clé-valeur où la clé est essentiellement une `string` et où la valeur est une `string`, un `z-set` ou une hmap.

Vous pouvez utiliser le AWS Glue Data Catalog pour créer un schéma et configurer des tables virtuelles. Des propriétés de table spéciales indiquent au connecteur Athena Redis comment mapper vos clés et valeurs Redis dans une table. Pour plus d'informations, consultez Configuration de bases de données et de tables dans AWS Glue dans la suite de ce document.

Si Lake Formation est activé sur votre compte, le rôle IAM de votre connecteur Lambda fédéré Athena que vous avez déployé dans AWS Serverless Application Repository le doit avoir un accès en lecture dans Lake Formation au. AWS Glue Data Catalog

Le connecteur Amazon Athena Redis prend en charge Amazon MemoryDB pour Redis et Amazon pour Redis. ElastiCache

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.
- Avant d'utiliser ce connecteur, configurez un VPC et un groupe de sécurité. Pour plus d'informations, consultez Création d'un VPC pour un connecteur de source de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Redis.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.
- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.
- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.
- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.

- glue_catalog – (Facultatif) Utilisez cette option pour spécifier un catalogue AWS Glue entre compte. Par défaut, le connecteur tente d'obtenir des métadonnées à partir de son propre AWS Glue compte.

Configuration de bases de données et de tables dans AWS Glue

Pour activer une AWS Glue table à utiliser avec Redis, vous pouvez définir les propriétés de table suivantes sur la table :`redis-endpoint`,`redis-value-type`, `redis-keys-zset` ou our`edis-key-prefix`.

En outre, toute AWS Glue base de données contenant des tables Redis doit avoir une propriété URI `redis-db-flag` dans la base de données. Pour définir la propriété `redis-db-flag` URI, utilisez la AWS Glue console pour modifier la base de données.

La liste suivante décrit les propriétés de la table.

- redis-endpoint — (Obligatoire) Le `:` *`mot de passe`* du `:` *`port`* du *`nom d'hôte`* du serveur Redis qui contient les données de cette table (par exemple,`athena-federation-demo.cache.amazonaws.com:6379`) Vous pouvez également stocker le point de terminaison, ou une partie du point de terminaison, en AWS Secrets Manager utilisant $ {*`secret_Name`*} comme valeur de propriété de table.

> **Note**
>
> Pour utiliser la fonctionnalité Athena Federated Query avec, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

- redis-keys-zset— (Obligatoire si elle n'`redis-key-prefix`est pas utilisée) Une liste de clés séparées par des virgules dont la valeur est un zset (par exemple,). `active-orders,pending-orders` Chacune des valeurs du zset est considérée comme une clé faisant partie de la table. La propriété `redis-keys-zset` ou `redis-key-prefix` doit être définie.
- redis-key-prefix— (Obligatoire si elle n'`redis-keys-zset`est pas utilisée) Une liste de préfixes clés séparés par des virgules pour rechercher des valeurs dans le tableau (par exemple,`accounts-*,acct-`). La propriété `redis-key-prefix` ou `redis-keys-zset` doit être définie.

- redis-value-type— (Obligatoire) Définit la façon dont les valeurs des clés définies par l'une `redis-key-prefix` ou l'autre `redis-keys-zset` sont mappées à votre table. Un littéral mappe à une seule colonne. Un zset mappe également à une seule colonne, mais chaque clé peut stocker de nombreuses lignes. Un hachage permet à chaque clé de constituer une ligne avec plusieurs colonnes (par exemple, un hachage, un littéral ou un zset).
- redis-ssl-flag— (Facultatif) Quand`True`, crée une connexion Redis utilisant SSL/TLS. L'argument par défaut est `False`.
- redis-cluster-flag— (Facultatif) Quand`True`, active la prise en charge des instances Redis en cluster. L'argument par défaut est `False`.
- redis-db-number— (Facultatif) S'applique uniquement aux instances autonomes non clusterisées.) Définissez ce nombre (par exemple 1, 2 ou 3) pour lire à partir d'une base de données Redis personnalisée. La base de données logique Redis par défaut est 0. Ce numéro ne fait pas référence à une base de données dans Athena ou AWS Glue, mais à une base de données logique Redis. Pour plus d'informations, consultez l'index SELECT dans la documentation Redis.

Types de données

Le connecteur Redis prend en charge les types de données suivants. Les flux Redis ne sont pas pris en charge.

- String
- Hachage
- Jeu trié (ZSet)

Toutes les valeurs Redis sont récupérées en tant que type de données `string`. Elles sont ensuite converties vers l'un des types de données Apache Arrow suivants en fonction de la façon dont vos tables sont définies dans le AWS Glue Data Catalog.

AWS Glue type de données	Type de données Apache Arrow
int	INT
chaîne	VARCHAR
bigint	BIGINT

AWS Glue type de données	Type de données Apache Arrow
double	FLOAT8
float	FLOAT4
smallint	SMALLINT
tinyint	TINYINT
boolean	BIT
binary	VARBINARY

Autorisations nécessaires

Pour plus de détails sur les politiques IAM requises par ce connecteur, reportez-vous à la section `Policies` du fichier athena-redis.yaml. La liste suivante résume les autorisations requises.

- Amazon S3 write access (Accès en écriture Amazon S3) – Le connecteur nécessite un accès en écriture à un emplacement dans Amazon S3 pour déverser les résultats à partir de requêtes volumineuses.
- Athena GetQueryExecution — Le connecteur utilise cette autorisation pour échouer rapidement lorsque la requête Athena en amont est terminée.
- AWS Glue Data Catalog— Le connecteur Redis nécessite un accès en lecture seule au pour AWS Glue Data Catalog obtenir des informations sur le schéma.
- CloudWatch Journaux : le connecteur a besoin d'accéder aux CloudWatch journaux pour stocker les journaux.
- AWS Secrets Manager accès en lecture — Si vous choisissez de stocker les détails du point de terminaison Redis dans Secrets Manager, vous devez autoriser le connecteur à accéder à ces secrets.
- Accès VPC – Le connecteur nécessite la possibilité d’attacher des interfaces à votre VPC et de les détacher afin qu’il puisse s’y connecter et communiquer avec vos instances Redis.

Performance

Le connecteur Athena Redis tente de paralléliser les requêtes par rapport à votre instance Redis en fonction du type de table que vous avez défini (par exemple, des clés zset ou des clés de préfixe).

Le connecteur Athena Redis effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Cependant, les requêtes contenant un prédicat contre la clé primaire échouent avec le délai d'attente. Les clauses `LIMIT` réduisent la quantité de données analysées, mais si vous ne fournissez pas de prédicat, vous devez vous attendre à ce que les requêtes `SELECT` avec une clause `LIMIT` analysent au moins 16 Mo de données. Le connecteur Redis résiste à la limitation due à la simultanéité.

Informations de licence

Le projet de connecteur Redis Amazon Athena est concédé sous licence dans le cadre de la licence Apache-2.0.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Redshift

Le connecteur Amazon Athena Redshift permet à Amazon Athena d'accéder à vos bases de données Amazon Redshift et Amazon Redshift Serverless, y compris les vues Redshift Serverless. Vous pouvez vous connecter à l'un ou l'autre service à l'aide des paramètres de configuration de la chaîne de connexion JDBC décrits sur cette page.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l’aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d’informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d’écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.

- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Comme Redshift ne prend pas en charge les partitions externes, toutes les données spécifiées par une requête sont récupérées à chaque fois.

Conditions

Les termes suivants se rapportent au connecteur Redshift :

- Instance de base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Redshift.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
redshift://${jdbc_connection_string}
```

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`RedshiftMuxCompositeHandler`
Gestionnaire de métadonnées	`RedshiftMuxMetadataHandler`
Gestionnaire d'enregis trements	`RedshiftMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
$*catalog*_connecti on_string	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `myredshiftcatalog` , le nom de la variable d'environnement est alors `myredshiftcatalog_ connection_string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *AWS_LAMBD A_FUNCTION_NAME* `}`.

Les exemples de propriétés suivants concernent une fonction Redshift MUX Lambda qui prend en charge deux instances de base de données : `redshift1` (par défaut) et `redshift2`.

Propriété	Valeur
`default`	`redshift://jdbc:redshift://redshift1.host:5439/dev?user=sample2&password=sample2`
`redshift_catalog1_connection_string`	`redshift://jdbc:redshift://redshift1.host:3306/default?${Test/RDS/Redshift1}`
`redshift_catalog2_connection_string`	`redshift://jdbc:redshift://redshift2.host:3333/default?user=sample2&password=sample2`

Fourniture des informations d’identification

Pour fournir un nom d’utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret ${Test/RDS/ `Redshift1`}.

```
redshift://jdbc:redshift://redshift1.host:3306/default?...&${Test/RDS/Redshift1}&...
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
redshift://jdbc:redshift://redshift1.host:3306/
default?...&user=sample2&password=sample2&...
```

Actuellement, le connecteur Redshift reconnaît les propriétés JDBC `user` et `password`.

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject`

Paramètre	Description
	Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant montre les types de données correspondants pour JDBC et Apache Arrow.

JDBC	Flèche
Booléen	Bit
Entier	Tiny
Court	Smallint
Entier	Int
Long	Bigint
float	Float4
Double	Float8
Date	DateDay
Horodatage	DateMilli
Chaîne	Varchar
Octets	Varbinary
BigDecimal	Décimal
ARRAY	Liste

Partitions et déversements

Redshift ne prend pas en charge les partitions externes. Pour plus d'informations sur les problèmes liés aux performances, veuillez consulter Performance.

Performance

Le connecteur Athena Redshift effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Les clauses `LIMIT`, les clauses `ORDER BY`, les prédicats simples et les expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution des requêtes. Cependant, la sélection d'un sous-ensemble de colonnes entraîne parfois un délai d'exécution plus long des requêtes. Amazon Redshift est particulièrement vulnérable au ralentissement de l'exécution des requêtes lorsque vous exécutez plusieurs requêtes en même temps.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Requêtes Top N

Une requête Top `N` spécifie le classement du jeu de résultats et la limite du nombre de lignes renvoyées. Vous pouvez utiliser ce type de requête pour déterminer les valeurs Top `N` maximales ou Top `N` minimales pour vos jeux de données. Grâce à la poussée vers le bas Top `N`, le connecteur renvoie uniquement des lignes `N` classées à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena Redshift peut combiner ces expressions et les pousser directement vers Redshift pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena Redshift suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL

- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%')
ORDER BY col_a DESC
LIMIT 10;
```

Pour un article sur l'utilisation de la poussée vers le bas de prédicats pour améliorer les performances des requêtes fédérées, y compris Amazon Redshift, consultez Améliorer les requêtes fédérées avec la poussée vers le bas de prédicats dans Amazon Athena sur le blog AWS Big Data.

Requêtes passthrough

Le connecteur Redshift prend en charge les requêtes passthrough. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Redshift, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Redshift. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Ressources supplémentaires

Pour obtenir les informations les plus récentes sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur Redshift sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour SAP HANA

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Dans SAP HANA, les noms d'objets sont convertis en majuscules lorsqu'ils sont stockés dans la base de données SAP HANA. Toutefois, comme les noms entre guillemets sont sensibles à la casse, il est possible que deux tables portent le même nom en minuscules et en majuscules (par exemple, `EMPLOYEE` et `employee`).

  Dans la requête fédérée d'Athena, les noms des tables de schéma sont fournis à la fonction Lambda en minuscules. Pour contourner ce problème, vous pouvez fournir des indicateurs de requête `@schemaCase` pour récupérer les données des tables dont les noms sont sensibles à la casse. Vous trouverez ci-dessous deux exemples de requêtes avec des indicateurs de requête.

```
SELECT *
FROM "lambda:saphanaconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=upper"
```

```
SELECT *
```

```
FROM "lambda:saphanaconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=lower"
```

Conditions

Les termes suivants se rapportent au connecteur SAP HANA :

- Instance de base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur SAP HANA.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
saphana://${jdbc_connection_string}
```

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`SaphanaMuxCompositeHandler`
Gestionnaire de métadonnées	`SaphanaMuxMetadataHandler`
Gestionnaire d'enregis trements	`SaphanaMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
$*catalog*_connecti on_string	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `mysaphanacatalog` , le nom de la variable d'environnement est alors `mysaphanacatalog_connection _string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *AWS_LAMBD A_FUNCTION_NAME* `}`.

Les exemples de propriétés suivants concernent une fonction Saphana MUX Lambda qui prend en charge deux instances de base de données : `saphana1` (par défaut) et `saphana2`.

Propriété	Valeur
`default`	`saphana://jdbc:sap://saphana1.host:port/?${Test/RDS/Saphana1}`
`saphana_catalog1_connection_string`	`saphana://jdbc:sap://saphana1.host:port/?${Test/RDS/Saphana1}`
`saphana_catalog2_connection_string`	`saphana://jdbc:sap://saphana2.host:port/?user=sample2&password=sample2`

Fourniture des informations d’identification

Pour fournir un nom d’utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${Test/RDS/Saphana1}`.

```
saphana://jdbc:sap://saphana1.host:port/?${Test/RDS/Saphana1}&...
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
saphana://jdbc:sap://saphana1.host:port/?user=sample2&password=sample2&...
```

Actuellement, le connecteur SAP HANA reconnaît les propriétés JDBC `user` et `password`.

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance SAP HANA.

Type de gestionnaire	Classe
Gestionnaire de composites	`SaphanaCompositeHandler`
Gestionnaire de métadonnées	`SaphanaMetadataHandler`
Gestionnaire d'enregis trements	`SaphanaRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une instance SAP HANA unique prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`saphana://jdbc:sap://saphana1.host:port/?secret=Test/RDS/Saphana1`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant montre les types de données correspondants pour JDBC et Apache Arrow.

JDBC	Flèche
Booléen	Bit
Entier	Tiny
Court	Smallint
Entier	Int
Long	Bigint
float	Float4
Double	Float8
Date	DateDay
Horodatage	DateMilli
Chaîne	Varchar
Octets	Varbinary
BigDecimal	Décimal
ARRAY	Liste

Conversions du type de données

Outre les conversions JDBC vers Arrow, le connecteur effectue certaines autres conversions pour que la source SAP HANA et les types de données Athena soient compatibles. Ces conversions permettent de garantir la réussite de l’exécution des requêtes. Le tableau suivant présente ces conversions.

Type de données source (SAP HANA)	Type de données converties (Athena)
DECIMAL	BIGINT
INTEGER	INT
DATE	DATEDAY
TIMESTAMP	DATEMILLI

Tous les autres types de données non pris en charge sont convertis en `VARCHAR`.

Partitions et déversements

Une partition est représentée par une seule colonne de partition de type `Integer`. La colonne contient les noms des partitions définies dans une table SAP HANA. Pour une table qui ne possède pas de nom de partition, * est renvoyé, ce qui équivaut à une partition unique. Une partition équivaut à une division.

Nom	Type	Description
PART_ID	Entier	Partition nommée dans SAP HANA.

Performance

SAP HANA prend en charge les partitions natives. Le connecteur Athena SAP HANA peut récupérer les données de ces partitions en parallèle. Si vous souhaitez interroger des jeux de données très volumineux avec une distribution de partition uniforme, le partitionnement natif est fortement recommandé. La sélection d'un sous-ensemble de colonnes accélère considérablement l'exécution des requêtes et réduit le nombre de données analysées. Le connecteur présente des limitations importantes et parfois des échecs de requêtes, en raison de la simultanéité.

Le connecteur Athena SAP HANA effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Les clauses `LIMIT`, les prédicats simples et les expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution des requêtes.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena SAP HANA peut combiner ces expressions et les pousser directement vers SAP HANA pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena SAP HANA suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%')
LIMIT 10;
```

Requêtes directes

Le connecteur SAP HANA prend en charge les requêtes directes. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec SAP HANA, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans SAP HANA. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les dernières informations sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur SAP HANA sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Snowflake

Le connecteur Amazon Athena pour Snowflake permet à Amazon Athena d'exécuter des requêtes SQL sur des données stockées dans vos bases de données Snowflake ou des instances RDS à l'aide de JDBC.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Pour l'instant, les affichages Snowflake à division unique sont pris en charge.
- Dans Snowflake, étant donné que les noms d'objets sont sensibles à la casse, deux tables peuvent porter le même nom en minuscules et en majuscules (par exemple, `EMPLOYEE` et `employee`). Dans la requête fédérée d'Athena, les noms des tables de schéma sont fournis à la fonction Lambda en minuscules. Pour contourner ce problème, vous pouvez fournir des indicateurs de requête `@schemaCase` pour récupérer les données des tables dont les noms sont sensibles à la casse. Vous trouverez ci-dessous deux exemples de requêtes avec des indicateurs de requête.

```
SELECT *
FROM "lambda:snowflakeconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=upper"
```

```
SELECT *
FROM "lambda:snowflakeconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=lower"
```

Conditions

Les termes suivants se rapportent au connecteur Snowflake.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.

- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Snowflake.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
snowflake://${jdbc_connection_string}
```

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`SnowflakeMuxCompositeHandler`
Gestionnaire de métadonnées	`SnowflakeMuxMetadataHandler`

Handler (Gestionnaire)	Classe
Gestionnaire d'enregis trements	`SnowflakeMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
`$`*`catalog`*`_connecti on_string`	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `mysnowflakecatalog` , le nom de la variable d'environnement est alors `mysnowflakecatalog _connection_string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *`AWS_LAMBD A_FUNCTION_NAME`* `}`.

Les exemples de propriétés suivants concernent une fonction Snowflake MUX Lambda qui prend en charge deux instances de base de données : `snowflake1` (par défaut) et `snowflake2`.

Propriété	Valeur
`default`	`snowflake://jdbc:snowflake://snowflake1.host: port/?warehouse=warehousename&db=db1&schema=s chema1&${Test/RDS/Snowflake1}`
`snowflake _catalog1 _connecti on_string`	`snowflake://jdbc:snowflake://snowflake1.host: port/?warehouse=warehousename&db=db1&schema=s chema1${Test/RDS/Snowflake1}`

Propriété	Valeur
`snowflake_catalog2_connection_string`	`snowflake://jdbc:snowflake://snowflake2.host:port/?warehouse=warehousename&db=db1&schema=schema1&user=sample2&password=sample2`

Fourniture des informations d’identification

Pour fournir un nom d’utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

  > **Important**
  >
  > Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

  Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n’utilise pas Amazon RDS, stockez les informations d’identification au format JSON au format suivant :

```
{"username": "${username}", "password": "${password}"}
```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${Test/RDS/Snowflake1}`.

```
snowflake://jdbc:snowflake://snowflake1.host:port/?
warehouse=warehousename&db=db1&schema=schema1${Test/RDS/Snowflake1}&...
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
snowflake://jdbc:snowflake://snowflake1.host:port/
warehouse=warehousename&db=db1&schema=schema1&user=sample2&password=sample2&...
```

À l'heure actuelle, Snowflake reconnaît les propriétés JDBC `user` et `password`. Il accepte également le nom d'utilisateur et le mot de passe au format *`nomutilisateur`*/*`motdepasse`* sans les clés `user` ou `password`.

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance Snowflake.

Type de gestionnaire	Classe
Gestionnaire de composites	`SnowflakeCompositeHandler`
Gestionnaire de métadonnées	`SnowflakeMetadataHandler`
Gestionnaire d'enregis trements	`SnowflakeRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une instance Snowflake unique prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`snowflake://jdbc:snowflake://snowflake1.host:port/?secret=Test/RDS/Snowflake1`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant montre les types de données correspondants pour JDBC et Apache Arrow.

JDBC	Flèche
Booléen	Bit
Entier	Tiny
Court	Smallint
Entier	Int
Long	Bigint
float	Float4
Double	Float8
Date	DateDay
Horodatage	DateMilli
Chaîne	Varchar
Octets	Varbinary
BigDecimal	Décimal
ARRAY	Liste

Conversions du type de données

Outre les conversions JDBC vers Arrow, le connecteur effectue certaines autres conversions pour que la source Snowflake et les types de données Athena soient compatibles. Ces conversions permettent de garantir la réussite de l’exécution des requêtes. Le tableau suivant présente ces conversions.

Type de données source (Snowflake)	Type de données converties (Athena)
TIMESTAMP	TIMESTAMPMILLI
DATE	TIMESTAMPMILLI
INTEGER	INT
DECIMAL	BIGINT
TIMESTAMP_NTZ	TIMESTAMPMILLI

Tous les autres types de données non pris en charge sont convertis en `VARCHAR`.

Partitions et déversements

Les partitions sont utilisées pour déterminer comment générer des divisions pour le connecteur. Athena construit une colonne synthétique de type `varchar` qui représente le schéma de partitionnement de la table afin d'aider le connecteur à générer des divisions. Le connecteur ne modifie pas la définition réelle de la table.

Pour créer cette colonne synthétique et les partitions, Athena a besoin de définir une clé primaire. Cependant, étant donné que Snowflake n'applique pas les contraintes de clé primaire, vous devez vous-même appliquer l'unicité. Si vous ne le faites pas, Athéna optera par défaut pour une seule division.

Performance

Pour des performances optimales, utilisez des filtres dans les requêtes dans la mesure du possible. En outre, nous recommandons vivement le partitionnement natif pour récupérer des jeux de données volumineux dont la distribution des partitions est uniforme. La sélection d'un sous-ensemble de colonnes accélère considérablement l'exécution des requêtes et réduit le nombre de données analysées. Le connecteur Snowflake résiste à la limitation due à la simultanéité.

Le connecteur Athena Snowflake effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Les clauses `LIMIT`, les prédicats simples et les expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution des requêtes.

Clauses LIMIT

Une instruction `LIMIT N` réduit les données analysées par la requête. Grâce à la poussée vers le bas `LIMIT N`, le connecteur renvoie uniquement des lignes `N` à Athena.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena Snowflake peut combiner ces expressions et les pousser directement vers Snowflake pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena Snowflake suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%')
LIMIT 10;
```

Requêtes directes

Le connecteur Snowflake prend en charge les requêtes passthrough. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes passthrough avec Snowflake, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Snowflake. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les dernières informations sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur Snowflake sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Microsoft SQL Server

Le connecteur Amazon Athena pour Microsoft SQL Server permet à Amazon Athena d'exécuter des requêtes SQL sur vos données stockées dans Microsoft SQL Server à l'aide de JDBC.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.

- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Dans des conditions de filtre, vous devez lancer les types de données `DateetTimestamp` vers le type de données approprié.
- Pour rechercher des valeurs négatives de type `Real` et `Float`, utilisez l'opérateur <= ou >=.
- Les types de données `binary`, `varbinary`, `image` et `rowversion` ne sont pas pris en charge.

Conditions

Les termes suivants se rapportent au connecteur SQL Server.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur SQL Server.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
sqlserver://${jdbc_connection_string}
```

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`SqlServerMuxCompositeHandler`
Gestionnaire de métadonnées	`SqlServerMuxMetadataHandler`
Gestionnaire d'enregis trements	`SqlServerMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
$*catalog*_connecti on_string	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `mysqlservercatalog` , le nom de la

Paramètre	Description
	variable d'environnement est alors `mysqlservercatalog_connection_string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${` *AWS_LAMBDA_FUNCTION_NAME* `}`.

Les exemples de propriétés suivants concernent une fonction Lambda SqlServer MUX qui prend en charge deux instances de base de données `sqlserver1` : (par défaut) et. `sqlserver2`

Propriété	Valeur
`default`	`sqlserver://jdbc:sqlserver://sqlserver1.` *hostname*`:`*port*`;databaseName=` *<database_name>* `;${`*secret1_name*`}`
`sqlserver_catalog1_connection_string`	`sqlserver://jdbc:sqlserver://sqlserver1.` *hostname*`:`*port*`;databaseName=` *<database_name>* `;${`*secret1_name*`}`
`sqlserver_catalog2_connection_string`	`sqlserver://jdbc:sqlserver://sqlserver2.` *hostname*`:`*port*`;databaseName=` *<database_name>* `;${`*secret2_name*`}`

Fourniture des informations d'identification

Pour fournir un nom d'utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

> **Important**
>
> Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

  Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

  ```
  {"username": "${username}", "password": "${password}"}
  ```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${secret_name}`.

```
sqlserver://jdbc:sqlserver://hostname:port;databaseName=<database_name>;${secret_name}
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
sqlserver://
jdbc:sqlserver://
hostname:port;databaseName=<database_name>;user=<user>;password=<password>
```

Utilisation d’un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d’enregistrements suivants pour vous connecter à une seule instance SQL Server.

Type de gestionnaire	Classe
Gestionnaire de composites	`SqlServerCompositeHandler`
Gestionnaire de métadonnées	`SqlServerMetadataHandler`
Gestionnaire d’enregis trements	`SqlServerRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L’exemple de propriété suivant concerne une instance SQL Server unique prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`sqlserver://jdbc:sqlserver://` *`hostname`*`:`*`port`*`;database Name=` *`<database_name>`* `;${`*`secret_name`*` }`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant montre les types de données correspondants pour SQL Server et Apache Arrow.

SQL Server	Flèche
bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT
int	INT
bigint	BIGINT
decimal	DECIMAL
numeric	FLOAT8
smallmoney	FLOAT8

SQL Server	Flèche
money	DECIMAL
float[24]	FLOAT4
float[53]	FLOAT8
real	FLOAT4
datetime	Date(MILLISECOND)
datetime2	Date(MILLISECOND)
smalldatetime	Date(MILLISECOND)
date	Date(DAY)
time	VARCHAR
datetimeoffset	Date(MILLISECOND)
char[n]	VARCHAR
varchar[n/max]	VARCHAR
nchar [n]	VARCHAR
nvarchar[n/max]	VARCHAR
text	VARCHAR
ntext	VARCHAR

Partitions et déversements

Une partition est représentée par une seule colonne de partition de type `varchar`. Dans le cas du connecteur SQL Server, une fonction de partition détermine la manière dont les partitions sont appliquées à la table. Les informations relatives à la fonction de partition et au nom de colonne sont extraites de la table de métadonnées de SQL Server. Une requête personnalisée obtient ensuite la partition. Les divisions sont créées en fonction du nombre de partitions distinctes reçues.

Performance

La sélection d'un sous-ensemble de colonnes accélère considérablement l'exécution des requêtes et réduit le nombre de données analysées. Le connecteur SQL Server résiste à la limitation due à la simultanéité.

Le connecteur Athena SQL Server effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Des prédicats simples et des expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution de la requête.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena SQL Server peut combiner ces expressions et les pousser directement vers SQL Server pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena SQL Server suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%');
```

Requêtes passthrough

Le connecteur SQL Server prend en charge les requêtes directes. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec SQL Server, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans SQL Server. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les informations les plus récentes sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur SQL Server sur GitHub .com.

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Teradata

Le connecteur Amazon Athena pour Teradata permet à Athena d'exécuter des requêtes SQL sur des données stockées dans vos bases de données Teradata.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un

connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Limites

- Les opérations DDL d'écriture ne sont pas prises en charge.
- Dans une configuration de multiplexeur, le compartiment de déversement et le préfixe sont partagés entre toutes les instances de base de données.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .

Conditions

Les termes suivants se rapportent au connecteur Teradata.

- Base de données – Toute instance de base de données déployée sur site, sur Amazon EC2 ou sur Amazon RDS.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`
- Gestionnaire de multiplexage – Un gestionnaire Lambda qui peut accepter et utiliser plusieurs connexions de base de données.

Prérequis de couche Lambda

Pour utiliser le connecteur Teradata avec Athena, vous devez créer une couche Lambda qui inclut le pilote Teradata JDBC. Une couche Lambda est une archive de fichier `.zip` qui contient du code supplémentaire pour une fonction Lambda. Lorsque vous déployez le connecteur Teradata sur votre compte, vous spécifiez l'ARN de la couche. Cela attache la couche Lambda avec le pilote Teradata JDBC au connecteur Teradata afin que vous puissiez l'utiliser avec Athena.

Pour davantage d'informations sur les couches Lambda, consultez la rubrique Création et partage de couches Lambda dans le Guide du développeur AWS Lambda .

Pour créer une couche Lambda pour le connecteur Teradata

1. Accédez à la page de téléchargement du pilote Teradata JDBC à l'adresse https://downloads.teradata.com/download/connectivity/jdbc-driver.
2. Téléchargez le pilote Teradata JDBC. Le site Web vous demande de créer un compte et d'accepter un contrat de licence pour télécharger le fichier.
3. Extrayez le fichier `terajdbc4.jar` à partir du fichier d'archive que vous avez téléchargé.
4. Créez la structure de dossier suivante et placez-y le fichier `.jar`.

   ```
   java\lib\terajdbc4.jar
   ```
5. Création d'un fichier `.zip` de l'ensemble de la structure de dossier qui contient le fichier `terajdbc4.jar`.
6. Connectez-vous à la AWS Lambda console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lambda/.
7. Choisissez Layers (Couches) dans le panneau de navigation, puis Create layer (Créer une couche).
8. Pour Name (Nom), saisissez un nom pour la couche (par exemple, `TeradataJava11LambdaLayer`).
9. Assurez-vous que l'option Upload a .zip file (Charger un fichier .zip) est sélectionnée.
10. Choisissez Upload (Charger), puis téléchargez le dossier compressé contenant le pilote Teradata JDBC.
11. Sélectionnez Create (Créer).
12. Sur la page de détails de la couche, copiez l'ARN du calque en choisissant l'icône du presse-papier en haut de la page.
13. Enregistrez l'ARN pour référence.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Teradata.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
teradata://${jdbc_connection_string}
```

Utilisation d'un gestionnaire de multiplexage

Vous pouvez utiliser un multiplexeur pour vous connecter à plusieurs instances de base de données à l'aide d'une seule fonction Lambda. Les demandes sont acheminées par nom de catalogue. Utilisez les classes suivantes dans Lambda.

Handler (Gestionnaire)	Classe
Gestionnaire de composites	`TeradataMuxCompositeHandler`
Gestionnaire de métadonnées	`TeradataMuxMetadataHandler`
Gestionnaire d'enregis trements	`TeradataMuxRecordHandler`

Paramètres du gestionnaire de multiplexage

Paramètre	Description
$*catalog*_connecti on_string	Obligatoire. Chaîne de connexion d'instance de base de données. Préfixez la variable d'environnement avec le nom du catalogue utilisé dans Athena. Par exemple, si le catalogue enregistré auprès d'Athena est `myteradatacatalog` , le nom de la

Paramètre	Description
	variable d'environnement est alors `myteradatacatalog_connection_string` .
`default`	Obligatoire. Chaîne de connexion par défaut. Cette chaîne est utilisée lorsque le catalogue est `lambda:${ AWS_LAMBDA_FUNCTION_NAME }`.

Les exemples de propriétés suivants concernent une fonction Teradata MUX Lambda qui prend en charge deux instances de base de données :`teradata1` (par défaut) et `teradata2`.

Propriété	Valeur
`default`	`teradata://jdbc:teradata://teradata2.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,user=sample2&password=sample2`
`teradata_catalog1_connection_string`	`teradata://jdbc:teradata://teradata1.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,${Test/RDS/Teradata1}`
`teradata_catalog2_connection_string`	`teradata://jdbc:teradata://teradata2.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,user=sample2&password=sample2`

Fourniture des informations d'identification

Pour fournir un nom d'utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

> **Important**
>
> Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

  Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

  ```
  {"username": "${username}", "password": "${password}"}
  ```

Exemple de chaîne de connexion avec un nom secret

La chaîne suivante porte le nom secret `${Test/RDS/Teradata1}`.

```
teradata://jdbc:teradata1.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,${Test/RDS/
Teradata1}&...
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
teradata://jdbc:teradata://teradata1.host/
TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,...&user=sample2&password=sample2&...
```

À l'heure actuelle, Teradata reconnaît les propriétés JDBC `user` et `password`. Il accepte également le nom d'utilisateur et le mot de passe au format *`nomutilisateur`*`/`*`motdepasse`* sans les clés `user` ou `password`.

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance Teradata.

Type de gestionnaire	Classe
Gestionnaire de composites	`TeradataCompositeHandler`
Gestionnaire de métadonnées	`TeradataMetadataHandler`
Gestionnaire d'enregis trements	`TeradataRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

L'exemple de propriété suivant concerne une instance Teradata unique prise en charge par une fonction Lambda.

Propriété	Valeur
`default`	`teradata://jdbc:teradata://teradata1.host/TMODE=ANSI,C`

Propriété	Valeur
	`HARSET=UTF8,DATABASE=TEST,secret=Test/RDS/Teradata1`

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant montre les types de données correspondants pour JDBC et Apache Arrow.

JDBC	Flèche
Booléen	Bit
Entier	Tiny
Court	Smallint
Entier	Int
Long	Bigint

JDBC	Flèche
float	Float4
Double	Float8
Date	DateDay
Horodatage	DateMilli
Chaîne	Varchar
Octets	Varbinary
BigDecimal	Décimal
ARRAY	Liste

Partitions et déversements

Une partition est représentée par une seule colonne de partition de type `Integer`. La colonne contient les noms des partitions définies dans une table Teradata. Pour une table qui ne possède pas de nom de partition, * est renvoyé, ce qui équivaut à une partition unique. Une partition équivaut à une division.

Nom	Type	Description
partition	Entier	Partition nommée dans Teradata.

Performance

Teradata prend en charge les partitions natives. Le connecteur Athena Teradata peut récupérer les données de ces partitions en parallèle. Si vous souhaitez interroger des jeux de données très volumineux avec une distribution de partition uniforme, le partitionnement natif est fortement recommandé. La sélection d'un sous-ensemble de colonnes ralentit considérablement l'exécution de la requête. Le connecteur présente une certaine limitation en raison de la simultanéité.

Le connecteur Athena Teradata effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Des prédicats simples et des expressions complexes sont poussés vers le connecteur afin de réduire la quantité de données analysées et le délai d'exécution de la requête.

Prédicats

Un prédicat est une expression contenue dans la clause `WHERE` d'une requête SQL qui prend une valeur booléenne et filtre les lignes en fonction de plusieurs conditions. Le connecteur Athena Teradata peut combiner ces expressions et les pousser directement vers Teradata pour améliorer la fonctionnalité et réduire la quantité de données analysées.

Les opérateurs du connecteur Athena Teradata suivants prennent en charge la poussée vers le bas de prédicats :

- Booléen : AND, OR, NOT
- Égalité : EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN_OR_EQUAL, NULL_IF, IS_NULL
- Arithmétique : ADD, SUBTRACT, MULTIPLY, DIVIDE, MODULUS, NEGATE
- Autres : LIKE_PATTERN, IN

Exemple de poussée combinée vers le bas

Pour améliorer les capacités de requête, combinez les types de poussée vers le bas, comme dans l'exemple suivant :

```
SELECT *
FROM my_table
WHERE col_a > 10
    AND ((col_a + col_b) > (col_c % col_d))
    AND (col_e IN ('val1', 'val2', 'val3') OR col_f LIKE '%pattern%');
```

Requêtes directes

Le connecteur Teradata prend en charge les requêtes passthrough. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes passthrough avec Teradata, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Teradata. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les dernières informations sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur Teradata sur .com. GitHub

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Timestream

Le connecteur Amazon Athena pour Timestream permet à Amazon Athena de communiquer avec Amazon Timestream, ce qui rend os données de séries temporelles accessibles via Amazon Timestream. Vous pouvez éventuellement l'utiliser AWS Glue Data Catalog comme source de métadonnées supplémentaires.

Amazon Timestream est une base de données de séries temporelles rapide, évolutive, entièrement gérée et spécialement conçue pour faciliter le stockage et l'analyse de milliers de milliards de points de données en séries chronologiques par jour. Timestream vous fait gagner du temps et réduire les coûts de gestion du cycle de vie des données en séries chronologiques en conservant les données récentes en mémoire et en déplaçant les données historiques vers un niveau de stockage à coût optimisé en fonction des politiques définies par l'utilisateur.

Si Lake Formation est activé sur votre compte, le rôle IAM de votre connecteur Lambda fédéré Athena que vous avez déployé dans AWS Serverless Application Repository le doit avoir un accès en lecture dans Lake Formation au. AWS Glue Data Catalog

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur Timestream.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.
- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.
- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.
- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.

- glue_catalog – (Facultatif) Utilisez cette option pour spécifier un catalogue AWS Glue entre compte. Par défaut, le connecteur tente d'obtenir des métadonnées à partir de son propre AWS Glue compte.

Configuration de bases de données et de tables dans AWS Glue

Vous pouvez éventuellement utiliser le AWS Glue Data Catalog comme source de métadonnées supplémentaires. Pour activer une AWS Glue table à utiliser avec Timestream, vous devez disposer d'une AWS Glue base de données et d'une table dont les noms correspondent à la base de données Timestream et à la table pour laquelle vous souhaitez fournir des métadonnées supplémentaires.

> **Note**
>
> Pour des performances optimales, n'utilisez que des minuscules pour vos noms de bases de données et de tables. L'utilisation d'une casse mixte oblige le connecteur à effectuer une recherche insensible à la casse, ce qui demande plus de temps de calcul.

Pour configurer AWS Glue une table à utiliser avec Timestream, vous devez définir ses propriétés dans. AWS Glue

Pour utiliser une AWS Glue table pour des métadonnées supplémentaires

1. Modifiez le tableau dans la AWS Glue console pour ajouter les propriétés de tableau suivantes :

   - timestream-metadata-flag— Cette propriété indique au connecteur Timestream que le connecteur peut utiliser la table pour des métadonnées supplémentaires. Vous pouvez fournir n'importe quelle valeur pour `timestream-metadata-flag` tant que la propriété `timestream-metadata-flag` est présente dans la liste des propriétés de la table.
   - _view_template – Lorsque vous utilisez AWS Glue pour des métadonnées supplémentaires, vous pouvez utiliser cette propriété de table et spécifier n'importe quel SQL Timestream en tant que vue. Le connecteur Athena Timestream utilise le SQL de la vue avec votre SQL d'Athena pour exécuter votre requête. Cela s'avère utile si vous souhaitez utiliser une fonctionnalité de Timestream SQL qui n'est pas disponible dans Athena.

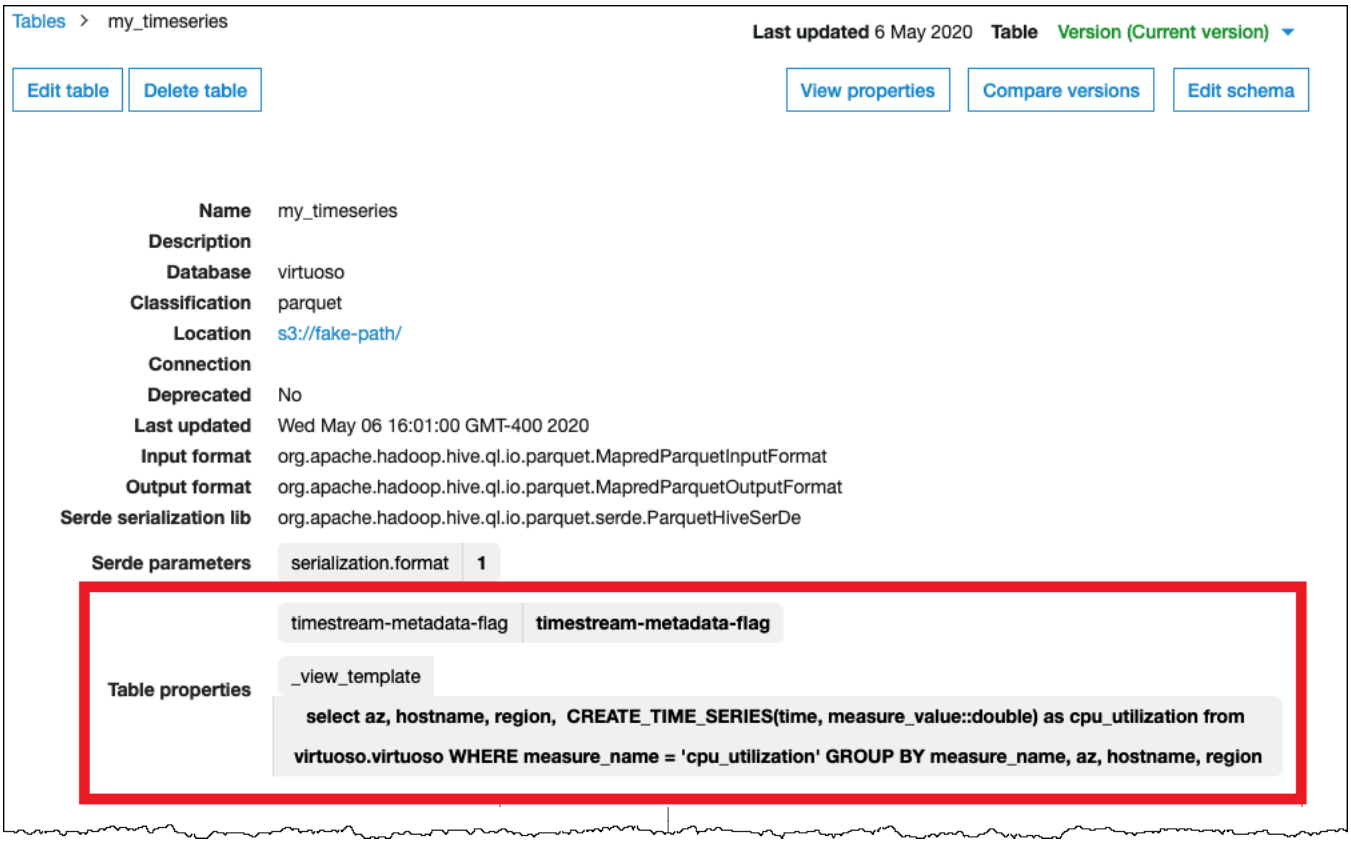
2. Assurez-vous d'utiliser les types de données appropriés AWS Glue tels que listés dans ce document.

Types de données

Actuellement, le connecteur Timestream ne prend en charge qu'un sous-ensemble des types de données disponibles dans Timestream, en particulier les valeurs scalaires `varchar`, `double` et `timestamp`.

Pour interroger le type de données `timeseries`, vous devez configurer une vue dans les propriétés de la table AWS Glue qui utilise la fonction `CREATE_TIME_SERIES` de Timestream. Vous devez également fournir un schéma pour la vue qui utilise la syntaxe `ARRAY<STRUCT<time:timestamp,measure_value::double:double>>` comme type pour n'importe laquelle de vos colonnes de séries chronologiques. Assurez-vous de remplacer `double` par le type scalaire approprié pour votre table.

L'image suivante montre un exemple de propriétés de AWS Glue table configurées pour configurer une vue sur une série chronologique.



Autorisations nécessaires

Pour plus de détails sur les politiques IAM requises par ce connecteur, reportez-vous à la section `Policies` du fichier athena-timestream.yaml. La liste suivante résume les autorisations requises.

- Amazon S3 write access (Accès en écriture Amazon S3) – Le connecteur nécessite un accès en écriture à un emplacement dans Amazon S3 pour déverser les résultats à partir de requêtes volumineuses.
- Athena GetQueryExecution — Le connecteur utilise cette autorisation pour échouer rapidement lorsque la requête Athena en amont est terminée.
- AWS Glue Data Catalog— Le connecteur Timestream nécessite un accès en lecture seule au pour AWS Glue Data Catalog obtenir des informations sur le schéma.
- CloudWatch Journaux : le connecteur a besoin d'accéder aux CloudWatch journaux pour stocker les journaux.
- Accès à Timestream – Pour exécuter des requêtes Timestream.

Performance

Nous vous recommandons d'utiliser la clause `LIMIT` afin de limiter les données renvoyées (et non les données numérisées) à moins de 256 Mo afin de garantir les performances des requêtes interactives.

Le connecteur Athena Timestream effectue une poussée vers le bas des prédicats pour réduire les données analysées par la requête. Les clauses `LIMIT` réduisent la quantité de données analysées, mais si vous ne fournissez pas de prédicat, vous devez vous attendre à ce que les requêtes `SELECT` avec une clause `LIMIT` analysent au moins 16 Mo de données. La sélection d'un sous-ensemble de colonnes accélère considérablement l'exécution des requêtes et réduit le nombre de données analysées. Le connecteur Timestream résiste à la limitation due à la simultanéité.

Requêtes directes

Le connecteur Timestream prend en charge les requêtes directes. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Timestream, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Timestream. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

Le projet de connecteur Timestream Amazon Athena est concédé sous licence dans le cadre de la licence Apache-2.0.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour TPC Benchmark DS (TPC-DS)

Le connecteur Amazon Athena pour TPC-DS permet à Amazon Athena de communiquer avec une source de données TPC Benchmark DS générées de manière aléatoire et destinées à être utilisées pour la définition de points de référence et les tests fonctionnels d'Athena Federation. Le connecteur TPC-DS d'Athena génère une base de données conforme TPC-DS à l'un des quatre facteurs d'échelle. Nous ne recommandons pas l'utilisation de ce connecteur comme alternative aux tests de performance des lacs de données basés sur Amazon S3.

Prérequis

- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.

Paramètres

Utilisez les variables d'environnement Lambda de cette section pour configurer le connecteur TPC-DS.

- spill_bucket – Spécifie le compartiment Amazon S3 pour les données qui dépassent les limites des fonctions Lambda.

- spill_prefix – (Facultatif) Par défaut, il s'agit d'un sous-dossier dans le `spill_bucket` spécifié appelé `athena-federation-spill`. Nous vous recommandons de configurer un cycle de vie de stockage Amazon S3 à cet endroit pour supprimer les déversements supérieurs à un nombre de jours ou d'heures prédéterminé.
- spill_put_request_headers – (Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObjectmanuel Amazon Simple Storage Service API Reference.
- kms_key_id – (Facultatif) Par défaut, toutes les données déversées vers Amazon S3 sont chiffrées à l'aide du mode de chiffrement authentifié AES-GCM et d'une clé générée de manière aléatoire. Pour que votre fonction Lambda utilise des clés de chiffrement plus puissantes générées par KMS, comme `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, vous pouvez spécifier l'ID d'une clé KMS.
- disable_spill_encryption – (Facultatif) Lorsque la valeur est définie sur `True`, le chiffrement des déversements est désactivé. La valeur par défaut est `False` afin que les données déversées vers S3 soient chiffrées à l'aide d'AES-GCM, soit à l'aide d'une clé générée de manière aléatoire soit à l'aide de KMS pour générer des clés. La désactivation du chiffrement des déversements peut améliorer les performances, surtout si votre lieu de déversement utilise le chiffrement côté serveur.

Tester des bases de données et des tables

Le connecteur TPC-DS d'Athena génère une base de données conforme TPC-DS à l'un des quatre facteurs d'échelle `tpcds1`, `tpcds10`, `tpcds100`, `tpcds250` ou `tpcds1000`.

Résumé des tables

Pour obtenir une liste complète des tables et des colonnes de données de test, exécutez les requêtes `SHOW TABLES` ou `DESCRIBE TABLE`. Le résumé des tables suivant est fourni pour des raisons de commodité.

1. call_center
2. catalog_page
3. catalog_returns
4. catalog_sales
5. customer
6. customer_address

7. customer_demographics
8. date_dim
9. dbgen_version
10. household_demographics
11. income_band
12. Inventory
13. item
14. promotion
15. raison
16. ship_mode
17. stocker
18. store_returns
19. store_sales
20. time_dim
21. warehouse
22. web_page
23. web_returns
24. web_sales
25. web_site

Pour les requêtes TPC-DS compatibles avec le schéma et les données générés, consultez le répertoire athena-tpcds/src/main/resources/queries/ sur. GitHub

Exemple de requête

Les exemples de requêtes `SELECT` suivants interrogent le catalogue `tpcds` pour la démographie des clients dans des comtés spécifiques.

```
SELECT
  cd_gender,
  cd_marital_status,
  cd_education_status,
  count(*) cnt1,
  cd_purchase_estimate,
  count(*) cnt2,
```

```
  cd_credit_rating,
  count(*) cnt3,
  cd_dep_count,
  count(*) cnt4,
  cd_dep_employed_count,
  count(*) cnt5,
  cd_dep_college_count,
  count(*) cnt6
FROM
  "lambda:tpcds".tpcds1.customer c, "lambda:tpcds".tpcds1.customer_address ca,
 "lambda:tpcds".tpcds1.customer_demographics
WHERE
  c.c_current_addr_sk = ca.ca_address_sk AND
    ca_county IN ('Rush County', 'Toole County', 'Jefferson County',
                  'Dona Ana County', 'La Porte County') AND
    cd_demo_sk = c.c_current_cdemo_sk AND
    exists(SELECT *
           FROM "lambda:tpcds".tpcds1.store_sales, "lambda:tpcds".tpcds1.date_dim
           WHERE c.c_customer_sk = ss_customer_sk AND
             ss_sold_date_sk = d_date_sk AND
             d_year = 2002 AND
             d_moy BETWEEN 1 AND 1 + 3) AND
    (exists(SELECT *
            FROM "lambda:tpcds".tpcds1.web_sales, "lambda:tpcds".tpcds1.date_dim
            WHERE c.c_customer_sk = ws_bill_customer_sk AND
              ws_sold_date_sk = d_date_sk AND
              d_year = 2002 AND
              d_moy BETWEEN 1 AND 1 + 3) OR
      exists(SELECT *
             FROM "lambda:tpcds".tpcds1.catalog_sales, "lambda:tpcds".tpcds1.date_dim
             WHERE c.c_customer_sk = cs_ship_customer_sk AND
               cs_sold_date_sk = d_date_sk AND
               d_year = 2002 AND
               d_moy BETWEEN 1 AND 1 + 3))
GROUP BY cd_gender,
  cd_marital_status,
  cd_education_status,
  cd_purchase_estimate,
  cd_credit_rating,
  cd_dep_count,
  cd_dep_employed_count,
  cd_dep_college_count
ORDER BY cd_gender,
  cd_marital_status,
```

```
  cd_education_status,
  cd_purchase_estimate,
  cd_credit_rating,
  cd_dep_count,
  cd_dep_employed_count,
  cd_dep_college_count
LIMIT 100
```

Autorisations nécessaires

Pour plus de détails sur les politiques IAM requises par ce connecteur, reportez-vous à la section `Policies` du fichier athena-tpcds.yaml. La liste suivante résume les autorisations requises.

- Amazon S3 write access (Accès en écriture Amazon S3) – Le connecteur nécessite un accès en écriture à un emplacement dans Amazon S3 pour déverser les résultats à partir de requêtes volumineuses.
- Athena GetQueryExecution — Le connecteur utilise cette autorisation pour échouer rapidement lorsque la requête Athena en amont est terminée.

Performance

Le connecteur Athena TPC-DS tente de paralléliser les requêtes en fonction du facteur d'échelle que vous choisissez. La poussée vers le bas de prédicat est effectuée au sein de la fonction Lambda.

Informations de licence

Le projet de connecteur TPC-DS Amazon Athena est concédé sous licence dans le cadre de la licence Apache-2.0.

Ressources supplémentaires

Pour plus d'informations sur ce connecteur, rendez-vous sur le site correspondant sur GitHub .com.

Connecteur Amazon Athena pour Vertica

Vertica est une plateforme de base de données en colonnes qui peut être déployée dans le cloud ou sur site et qui prend en charge les entrepôts de données à l'échelle des exaoctets. Vous pouvez utiliser le connecteur Vertica d'Amazon Athena dans les requêtes fédérées pour interroger les sources de données Vertica depuis Athena. Par exemple, vous pouvez exécuter des requêtes analytiques sur un entrepôt de données sur Vertica et un lac de données sur Simple Storage Service (Amazon S3).

Prérequis

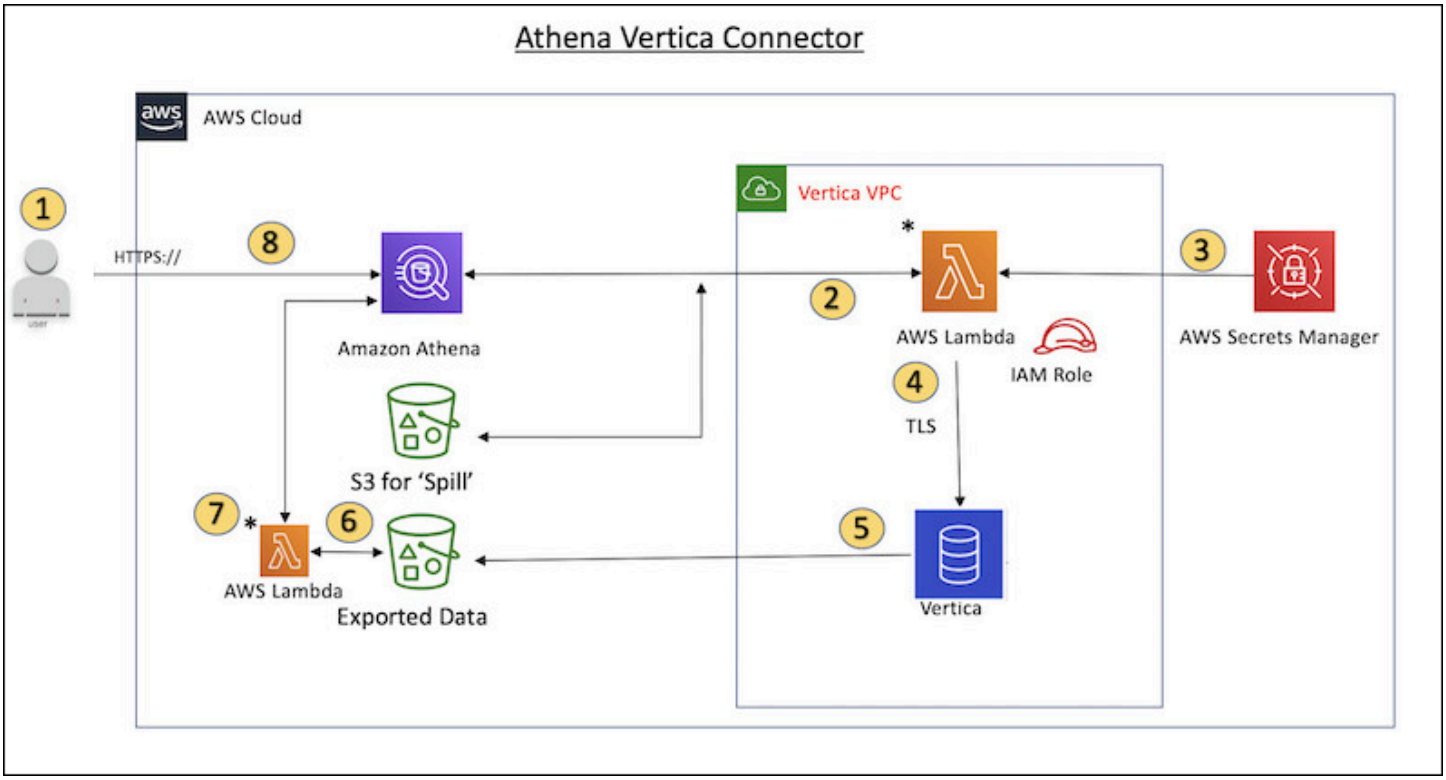
- Déployez le connecteur sur votre Compte AWS à l'aide de la console Athena ou du AWS Serverless Application Repository. Pour plus d'informations, consultez Déploiement d'un connecteur de source de données ou Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données.
- Avant d'utiliser ce connecteur, configurez un VPC et un groupe de sécurité. Pour plus d'informations, consultez Création d'un VPC pour un connecteur de source de données.

Limites

- Étant donné que le connecteur Athena utilise Amazon S3 Select pour lire des fichiers Parquet depuis Amazon S3, les performances du connecteur peuvent être lentes. Lorsque vous interrogez des tables de grande taille, nous vous recommandons d'utiliser une requête CREATE TABLE AS (SELECT ...) et des prédicats SQL.
- Actuellement, en raison d'un problème connu dans la requête fédérée d'Athena, le connecteur oblige Vertica à exporter toutes les colonnes de la table interrogée vers Amazon S3, mais seules les colonnes demandées sont visibles dans les résultats sur la console Athena.
- Les opérations DDL d'écriture ne sont pas prises en charge.
- Toutes les limites Lambda pertinentes. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .

Flux de travail

Le diagramme suivant montre le flux de travail d'une requête qui utilise le connecteur Vertica.



1. Une requête SQL est émise par rapport à une ou plusieurs tables dans Vertica.
2. Le connecteur analyse la requête SQL pour envoyer la partie appropriée à Vertica via la connexion JDBC.
3. Les chaînes de connexion utilisent le nom d'utilisateur et le mot de passe enregistrés AWS Secrets Manager pour accéder à Vertica.
4. Le connecteur enveloppe la requête SQL avec une commande EXPORT Vertica, comme dans l'exemple suivant.

```
EXPORT TO PARQUET (directory = 's3://DOC-EXAMPLE-BUCKET/folder_name',
   Compression='Snappy', fileSizeMB=64) OVER() as
SELECT
PATH_ID,
...
SOURCE_ITEMIZED,
SOURCE_OVERRIDE
FROM DELETED_OBJECT_SCHEMA.FORM_USAGE_DATA
WHERE PATH_ID <= 5;
```

5. Vertica traite la requête SQL et envoie l'ensemble de résultats à un compartiment Amazon S3. Pour un meilleur débit, Vertica utilise l'option `EXPORT` pour paralléliser l'opération d'écriture de plusieurs fichiers Parquet.
6. Athena analyse le compartiment Amazon S3 afin de déterminer le nombre de fichiers à lire pour l'ensemble de résultats.
7. Athena effectue plusieurs appels à la fonction Lambda et utilise Amazon S3 Select pour lire les fichiers Parquet à partir de l'ensemble de résultats. Les appels multiples permettent à Athena de paralléliser la lecture des fichiers Amazon S3 et d'obtenir un débit pouvant atteindre 100 Go par seconde.
8. Athena traite les données renvoyées par Vertica avec les données analysées depuis le lac de données et renvoie le résultat.

Conditions

Les termes suivants se rapportent au connecteur Vertica.

- Base de données – Toute instance d'une base de données Vertica déployée sur Amazon EC2.
- Gestionnaire – Un gestionnaire Lambda qui accède à votre instance de base de données. Un gestionnaire peut être destiné aux métadonnées ou aux enregistrements de données.
- Gestionnaire de métadonnées – Un gestionnaire Lambda qui extrait les métadonnées de votre instance de base de données.
- Gestionnaire d'enregistrements – Un gestionnaire Lambda qui extrait les enregistrements de données de votre instance de base de données.
- Gestionnaire de composites – Un gestionnaire Lambda qui extrait les métadonnées et les enregistrements de données de votre instance de base de données.
- Propriété ou paramètre – Propriété de base de données utilisée par les gestionnaires pour extraire des informations de base de données. Vous configurez ces propriétés en tant que variables d'environnement Lambda.
- Chaîne de connexion – Chaîne de texte utilisée pour établir une connexion à une instance de base de données.
- Catalogue — Un AWS Glue non-catalogue enregistré auprès d'Athena qui est un préfixe obligatoire pour la propriété. `connection_string`

Paramètres

Le connecteur Vertica d'Amazon Athena expose des options de configuration par le biais de variables d'environnement Lambda. Vous pouvez utiliser les variables d'environnement Lambda suivantes pour configurer le connecteur.

- AthenaCatalogName— Nom de la fonction Lambda
- ExportBucket— Le compartiment Amazon S3 dans lequel les résultats des requêtes Vertica sont exportés.
- SpillBucket— Le nom du compartiment Amazon S3 dans lequel cette fonction peut diffuser des données.
- SpillPrefix— Le préfixe de l'`SpillBucket`emplacement où cette fonction peut diffuser des données.
- SecurityGroupIds— Un ou plusieurs identifiants correspondant au groupe de sécurité qui doit être appliqué à la fonction Lambda (par exemple `sg1sg2`, ou`sg3`).
- SubnetIds— Un ou plusieurs ID de sous-réseau correspondant au sous-réseau que la fonction Lambda peut utiliser pour accéder à votre source de données (par exemple, `subnet1` ou). `subnet2`
- SecretNameOrPrefix— Le nom ou le préfixe d'un ensemble de noms dans Secrets Manager auquel cette fonction a accès (par exemple,`vertica-*`)
- VerticaConnectionString— Les détails de connexion Vertica à utiliser par défaut si aucune connexion spécifique au catalogue n'est définie. La chaîne peut éventuellement utiliser AWS Secrets Manager la syntaxe (par exemple,`${secret_name}`).
- ID de VPC – L'ID de VPC à associer à la fonction Lambda.

Chaîne de connexion

Utilisez une chaîne de connexion JDBC au format suivant pour vous connecter à une instance de base de données.

```
vertica://jdbc:vertica://host_name:port/database?user=vertica-
username&password=vertica-password
```

Utilisation d'un gestionnaire de connexion unique

Vous pouvez utiliser les métadonnées de connexion unique et les gestionnaires d'enregistrements suivants pour vous connecter à une seule instance Vertica.

Type de gestionnaire	Classe
Gestionnaire de composites	`VerticaCompositeHandler`
Gestionnaire de métadonnées	`VerticaMetadataHandler`
Gestionnaire d’enregis trements	`VerticaRecordHandler`

Paramètres du gestionnaire de connexion unique

Paramètre	Description
`default`	Obligatoire. Chaîne de connexion par défaut.

Les gestionnaires de connexion unique prennent en charge une instance de base de données et doivent fournir un paramètre de connexion `default`. Toutes les autres chaînes de connexion sont ignorées.

Fourniture des informations d’identification

Pour fournir un nom d’utilisateur et un mot de passe pour votre base de données dans votre chaîne de connexion JDBC, vous pouvez utiliser les propriétés de la chaîne de connexion ou AWS Secrets Manager.

- Chaîne de connexion – Un nom d'utilisateur et un mot de passe peuvent être spécifiés en tant que propriétés dans la chaîne de connexion JDBC.

> **Important**
>
> Afin de vous aider à optimiser la sécurité, n'utilisez pas d'informations d'identification codées en dur dans vos variables d'environnement ou vos chaînes de connexion. Pour plus d'informations sur le transfert de vos secrets codés en dur vers AWS Secrets

Manager, voir Déplacer les secrets codés en dur vers AWS Secrets Manager dans le Guide de l'AWS Secrets Manager utilisateur.

- AWS Secrets Manager— Pour utiliser la fonctionnalité Athena Federated Query, AWS Secrets Manager le VPC connecté à votre fonction Lambda doit disposer d'un accès Internet ou d'un point de terminaison VPC pour se connecter à Secrets Manager.

  Vous pouvez insérer le nom d'un secret AWS Secrets Manager dans votre chaîne de connexion JDBC. Le connecteur remplace le nom secret par les valeurs `username` et `password` de Secrets Manager.

  Pour les instances de base de données Amazon RDS, cette prise en charge est étroitement intégrée. Si vous utilisez Amazon RDS, nous vous recommandons vivement d'utiliser une AWS Secrets Manager rotation des identifiants. Si votre base de données n'utilise pas Amazon RDS, stockez les informations d'identification au format JSON au format suivant :

  ```
  {"username": "${username}", "password": "${password}"}
  ```

Exemple de chaîne de connexion avec des noms secrets

La chaîne suivante contient les noms secrets `${vertica-username}` et `${vertica-password}`.

```
vertica://jdbc:vertica://host_name:port/database?user=${vertica-username}&password=
${vertica-password}
```

Le connecteur utilise le nom secret pour récupérer les secrets et fournir le nom d'utilisateur et le mot de passe, comme dans l'exemple suivant.

```
vertica://jdbc:vertica://host_name:port/database?user=sample-user&password=sample-
password
```

Actuellement, le connecteur Vertica reconnaît les propriétés JDBC `vertica-username` et `vertica-password`.

Paramètres de déversement

Le kit SDK Lambda peut déverser des données vers Amazon S3. Toutes les instances de base de données accessibles par la même fonction Lambda déversent au même emplacement.

Paramètre	Description
`spill_bucket`	Obligatoire. Nom du compartiment de déversement.
`spill_prefix`	Obligatoire. Préfixe de la clé du compartiment de déversement.
`spill_put_request_headers`	(Facultatif) Une carte codée au format JSON des en-têtes et des valeurs des demandes pour la demande `putObject` Amazon S3 utilisée pour le déversement (par exemple, `{"x-amz-server-side-encryption" : "AES256"}`). Pour les autres en-têtes possibles, consultez le PutObject manuel Amazon Simple Storage Service API Reference.

Prise en charge du type de données

Le tableau suivant indique les types de données pris en charge pour le connecteur Vertica.

Booléen
BigInt
Court
Entier
Long
Float
Double
Date
Varchar
Octets
BigDecimal

Booléen
TimeStamp dans le rôle de : Varchar

Performance

La fonction Lambda effectue une poussée vers le bas de projection pour réduire les données analysées par la requête. Les clauses `LIMIT` réduisent la quantité de données analysées, mais si vous ne fournissez pas de prédicat, vous devez vous attendre à ce que les requêtes `SELECT` avec une clause `LIMIT` analysent au moins 16 Mo de données. Le connecteur Vertica résiste à la limitation de à la simultanéité.

Requêtes passthrough

Le connecteur Vertica prend en charge les requêtes directes. Les requêtes passthrough utilisent une fonction de table pour transférer votre requête complète vers la source de données pour exécution.

Pour utiliser des requêtes directes avec Vertica, vous pouvez utiliser la syntaxe suivante :

```
SELECT * FROM TABLE(
        system.query(
            query => 'query string'
        ))
```

L'exemple de requête suivant envoie une requête vers une source de données dans Vertica. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informations de licence

En utilisant ce connecteur, vous reconnaissez l'inclusion de composants tiers, dont la liste se trouve dans le fichier pom.xml de ce connecteur, et vous acceptez les termes des licences tierces respectives fournies dans le fichier LICENSE.txt sur GitHub .com.

Ressources supplémentaires

Pour obtenir les informations les plus récentes sur la version du pilote JDBC, consultez le fichier pom.xml du connecteur Vertica sur .com. GitHub

Pour plus d'informations sur ce connecteur, consultez le site correspondant sur GitHub .com et Interrogation d'une source de données Vertica dans Amazon Athena à l'aide du SDK Athena Federated Query sur le blog Big Data.AWS

Déploiement d'un connecteur de source de données

La préparation à la création de requêtes fédérées est un processus en deux parties : le déploiement d'un connecteur de source de données de fonction Lambda et la connexion de la fonction Lambda à une source de données. Dans ce processus, vous donnez à la fonction Lambda un nom que vous pouvez choisir ultérieurement dans la console Athena et donnez au connecteur un nom que vous pouvez référencer dans vos requêtes SQL.

> **Note**
>
> Pour utiliser la fonctionnalité Athena Federated Query avec AWS Secrets Manager, vous devez configurer un point de terminaison privé Amazon VPC pour Secrets Manager. Pour plus d'informations, consultez la rubrique Création d'un point de terminaison privé VPC Secrets Manager du Guide de l'utilisateur AWS Secrets Manager .

Rubriques

- Utilisation de la console Athena
- Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données
- Création d'un VPC pour un connecteur de source de données
- Activation des requêtes fédérées entre comptes
- Mise à jour d'un connecteur de source de données

Utilisation de la console Athena

Pour choisir, nommer et déployer un connecteur de source de données, vous utilisez les consoles Athena et Lambda dans un processus intégré.

Pour déployer un connecteur de source de données

1. Ouvrez la console Athena à l’adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le panneau de navigation, choisissez Sources de données.
4. À la page Data sources (Sources de données), choisissez Connect data source (Connecter la source de données).
5. Pour Choose a data source (Sélectionnez une source de données), choisissez la source de données qu'Athena doit interroger, en tenant compte des directives suivantes :

   - Choisissez une option de requête fédérée qui correspond à votre source de données. Athena dispose de connecteurs de source de données prédéfinis que vous pouvez configurer pour des sources comme MySQL, Amazon DocumentDB et PostgreSQL.
   - Choisissez S3 - AWS Glue Data Catalog si vous souhaitez interroger des données dans Simple Storage Service (Amazon S3) et que vous n'utilisez pas un métastore Apache Hive ou l'une des autres options de source de données de requête fédérée de cette page. Athena utilise le AWS Glue Data Catalog pour stocker les métadonnées et les informations de schéma pour les sources de données Simple Storage Service (Amazon S3). Il s'agit de l'option par défaut (non fédérée). Pour plus d’informations, consultez Utilisation AWS Glue pour se connecter à des sources de données dans Amazon S3.
   - Choisissez S3 - Apache Hive metastore (métastore Apache Hive) pour interroger les jeux de données dans Simple Storage Service (Amazon S3) qui utilisent un métastore Apache

Hive. Pour plus d'informations sur cette option, consultez Connexion d'Athena à un métastore Apache Hive.

- Choisissez Custom or shared connector (Connecteur personnalisé ou partagé) si vous souhaitez créer votre propre connecteur de source de données à utiliser avec Athena. Pour plus d'informations sur l'écriture d'un connecteur de source de données , consultez Développement d'un connecteur de source de données à l'aide du kit SDK Athena Query Federation.

Ce didacticiel choisit Amazon CloudWatch Logs comme source de données fédérée.

6. Choisissez Suivant.
7. À la page Enter data source details (Saisir les détails de la source de données), pour Data source name (Nom de la source de données), saisissez le nom que vous souhaitez utiliser dans vos instructions SQL lorsque vous interrogez la source de données à partir d'Athena (par exemple, `CloudWatchLogs`). Le nom peut contenir jusqu'à 127 caractères et doit être unique dans votre compte. Il ne peut pas être modifié après sa création. Les caractères valides sont a-z, A-Z, 0–9, _ (trait de soulignement), @ (arobase) et - (trait d'union). Les noms `awsdatacatalog`, `hive`, `jmx` et `system` sont réservés par Athena et ne peuvent pas être utilisés pour les noms de source de données.
8. Pour Lambda Function (fonction Lambda), choisissez Create Lambda Function (Créer une fonction Lambda). La page de fonctions du connecteur que vous avez choisi s'ouvre dans la AWS Lambda console. La page contient des informations détaillées sur le connecteur.
9. Sous Application settings (Paramètres d'application), lisez attentivement la description de chaque paramètre d'application, puis saisissez les valeurs qui correspondent à vos besoins.

   Les paramètres de l'application que vous voyez varient en fonction du connecteur de votre source de données. Les paramètres minimaux requis sont les suivants :

   - AthenaCatalogName— Nom, en minuscules, de la fonction Lambda qui indique la source de données qu'elle cible, par exemple. `cloudwatchlogs`
   - SpillBucket— Un compartiment Amazon S3 dans votre compte pour stocker les données qui dépassent les limites de taille de réponse de la fonction Lambda.

   > **Note**
   >
   > Les données déversées ne sont pas réutilisées lors d'exécutions ultérieures et peuvent être supprimées en toute sécurité après 12 heures. Athena ne supprime pas ces

données pour vous. Pour gérer ces objets, envisagez d'ajouter une politique de cycle de vie des objets qui supprime les anciennes données de votre compartiment de déversement Simple Storage Service (Amazon S3). Pour plus d'informations, veuillez consulter Gestion du cycle de vie des objets dans le Guide de l'utilisateur Amazon S3.

10. Sélectionnez I acknowledge that this app creates custom IAM roles and resource policies (Je reconnais que cette application crée des politiques de ressources et rôles IAM personnalisés). Pour de plus amples informations, veuillez cliquer sur le lien Info.
11. Choisissez Deploy (Déployer). Lorsque le déploiement est terminé, la fonction Lambda apparaît dans la section Resources (Ressources) dans la console Lambda.

Connexion à une source de données

Une fois que vous avez déployé le connecteur de source de données sur votre compte, vous pouvez y connecter Athena.

Connexion d'Athena à une source de données à l'aide d'un connecteur que vous avez déployé sur votre compte

1. Revenez à la page Enter data sources details (Saisir les détails des sources de données) de la console Athena.
2. Dans la section Connection details (Détails de connexion), choisissez l'icône actualiser à côté de la zone de recherche Select or enter a Lambda function (Sélectionner ou saisir une fonction Lambda.
3. Choisissez le nom de la fonction que vous venez de créer dans la console Lambda. L'ARN de la fonction Lambda s'affiche.
4. (Facultatif) Pour Tags (Identifications), ajoutez des paires clé-valeur à associer à cette source de données. Pour en savoir plus sur les identifications, consultez Étiquetage des ressources Athena.
5. Choisissez Suivant.
6. À la page Review and create, vérifiez les détails de la source de données, puis choisissez Create data source (Créer une source de données).
7. La section Data source details (Détails de source de données) de la page de votre source de données affiche des informations sur votre nouveau connecteur. Vous pouvez maintenant utiliser le connecteur dans vos requêtes Athena.

Pour de plus amples informations sur l'utilisation de connecteurs de données dans les requêtes, consultez Exécution de requêtes fédérées.

Utilisation de AWS Serverless Application Repository pour déployer un connecteur de source de données

Pour déployer un connecteur de source de données, vous pouvez utiliser le AWS Serverless Application Repository au lieu de commencer par la console Athena. Utilisez le AWS Serverless Application Repository pour trouver le connecteur que vous souhaitez utiliser, fournissez les paramètres requis par le connecteur, puis déployez le connecteur sur votre compte. Ensuite, après avoir déployé le connecteur, vous utilisez la console Athena pour rendre la source de données disponible pour Athena.

Déploiement du connecteur sur votre compte

Pour utiliser l'AWS Serverless Application Repository pour déployer un connecteur de source de données sur votre compte

1. Connectez-vous à AWS Management Console et ouvrez le référentiel d'applications sans serveur.
2. Dans le volet de navigation, choisissez Applications.
3. Sélectionnez l'option Show apps that create custom IAM roles or resource policies (Afficher les applications qui créent des rôles IAM ou des politiques de ressources personnalisés).
4. Dans la zone de recherche, saisissez le nom du connecteur. Pour obtenir la liste des connecteurs de données Athena prédéfinis, voir Connecteurs de source de données disponibles.
5. Choisissez le nom du connecteur. Cliquer sur un connecteur ouvre la page Application details (Détails de l'application) de la fonction Lambda dans la console AWS Lambda.
6. Sur le côté droit de la page de détails, pour Application settings (Paramètres de l'application), fournissez les informations requises. Les paramètres minimaux requis sont les suivants. Pour plus d'informations sur les options configurables restantes pour les connecteurs de données créés par Athena, consultez la rubrique Connecteurs disponibles correspondante sur GitHub.

   - AthenaCatalogName – Nom de la fonction Lambda en minuscules qui indique la source de données qui est ciblée, par exemple `cloudwatchlogs`.

- SpillBucket : spécifiez un compartiment Simple Storage Service (Amazon S3) dans votre compte pour recevoir les données provenant de volumineuses charges utiles de réponse supérieures aux limites de taille de réponse de la fonction Lambda.

7. Sélectionnez I acknowledge that this app creates custom IAM roles and resource policies (Je reconnais que cette application crée des politiques de ressources et rôles IAM personnalisés). Pour de plus amples informations, veuillez cliquer sur le lien Info.
8. En bas à droite de la section Application settings (Paramètres de l'application), choisissez Deploy (Déployer). Lorsque le déploiement est terminé, la fonction Lambda apparaît dans la section Resources (Ressources) dans la console Lambda.

Mise à disposition du connecteur dans Athena

Maintenant vous pouvez utiliser la console Athena pour rendre le connecteur de source de données disponible pour Athena.

Pour rendre le connecteur de source de données disponible pour Athena

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le panneau de navigation, choisissez Sources de données.
4. À la page Data sources (Sources de données), choisissez Connect data source (Connecter la source de données).

5. Pour Choose a data source (Choisisir une source de données), choisissez la source de données pour laquelle vous avez créé un connecteur dans le AWS Serverless Application Repository. Ce tutoriel utilise Amazon CloudWatch Logs comme source de données fédérée.
6. Choisissez Suivant.
7. À la page Enter data source details (Saisir les détails de la source de données), pour Data source name (Nom de la source de données), saisissez le nom que vous souhaitez utiliser dans vos instructions SQL lorsque vous interrogez la source de données à partir d'Athena (par exemple, `CloudWatchLogs`). Le nom peut contenir jusqu'à 127 caractères et doit être unique dans votre compte. Il ne peut pas être modifié après sa création. Les caractères valides sont a-z, A-Z, 0–9, _ (trait de soulignement), @ (arobase) et - (trait d'union). Les noms `awsdatacatalog`, `hive`, `jmx` et `system` sont réservés par Athena et ne peuvent pas être utilisés pour les noms de source de données.
8. Dans la section Connection details (Détails de connexion), utilisez la zone Select or enter a Lambda function (Sélectionner ou saisir une fonction Lambda) pour choisir le nom de la fonction que vous venez de créer. L'ARN de la fonction Lambda s'affiche.
9. (Facultatif) Pour Tags (Identifications), ajoutez des paires clé-valeur à associer à cette source de données. Pour en savoir plus sur les identifications, consultez Étiquetage des ressources Athena.
10. Choisissez Suivant.
11. À la page Review and create, vérifiez les détails de la source de données, puis choisissez Create data source (Créer une source de données).
12. La section Data source details (Détails de source de données) de la page de votre source de données affiche des informations sur votre nouveau connecteur. Vous pouvez maintenant utiliser le connecteur dans vos requêtes Athena.

    Pour de plus amples informations sur l'utilisation de connecteurs de données dans les requêtes, consultez Exécution de requêtes fédérées.

Création d'un VPC pour un connecteur de source de données

Certains connecteurs de source de données Athena nécessitent un VPC et un groupe de sécurité. Cette rubrique explique comment créer un VPC avec un sous-réseau et un groupe de sécurité pour le VPC. Dans le cadre de ce processus, vous récupérez les ID du VPC, du sous-réseau et du groupe de sécurité que vous créez. Ces identifiants sont requis lorsque vous configurez votre connecteur pour une utilisation avec Athena.

Pour créer un VPC pour un connecteur de source de données Athena

1. Connectez-vous à la console Amazon VPC AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/vpc/.
2. Sélectionnez Create VPC (Créer un VPC).
3. Sur la page Créer un VPC, sous Paramètres VPC, dans Ressources à créer, sélectionnez VPC et plus encore.
4. Sous Génération automatique des étiquettes nominatives, dans le champ Génération automatique, entrez une valeur qui sera utilisée pour générer des étiquettes nominatives pour toutes les ressources de votre VPC.
5. Sélectionnez Create VPC (Créer un VPC).
6. Lorsque le processus est terminé, choisissez View VPC.
7. Dans la section Details (Détails), pour VPC ID (ID du VPC), copiez votre ID VPC pour référence ultérieure.

Vous pouvez maintenant récupérer l'ID de sous-réseau du VPC que vous venez de créer.

Pour récupérer votre ID de sous-réseau de VPC

1. Dans le panneau de navigation de la console VPC, choisissez Subnets (Sous-réseaux).
2. Sélectionnez le nom d'un sous-réseau dont la colonne VPC possède l'ID VPC que vous avez noté.
3. Dans la section Details (Détails), pour Subnet ID (ID du sous-réseau), copiez votre ID de sous-réseau pour référence ultérieure.

Ensuite, créez un groupe de sécurité pour votre VPC.

Pour créer un groupe de sécurité pour votre VPC

1. Dans le panneau de navigation de la console VPC, choisissez Security (Sécurité), Security Groups (Groupes de sécurité).
2. Sélectionnez Create security group (Créer un groupe de sécurité).
3. Sur la page Create Security Group (Créer un groupe de sécurité), saisissez les informations suivantes :

- Pour Security group name (Nom du groupe de sécurité), saisissez un nom pour votre groupe de sécurité.
- Pour Description, saisissez une description du groupe de sécurité. Une description est requise.
- Pour VPC, choisissez l'ID VPC du VPC que vous avez créé pour votre connecteur de source de données.
- Pour Inbound rules (Règles entrantes) et Outbound rules (Règles sortantes), ajoutez toutes les règles entrantes et sortantes dont vous avez besoin.

4. Sélectionnez Create security group (Créer un groupe de sécurité).
5. Sur la page Details (Details) du groupe de sécurité, copiez le Security group ID (ID du groupe de sécurité) pour référence ultérieure.

Activation des requêtes fédérées entre comptes

La requête fédérée vous permet d'interroger des sources de données autres qu'Amazon S3 à l'aide de connecteurs de source de données déployés sur AWS Lambda. La fonction de requête fédérée entre comptes permet à la fonction Lambda et aux sources de données à interroger d'être localisées dans différents comptes.

En tant qu'administrateur de données, vous pouvez activer les requêtes fédérées entre comptes en partageant votre connecteur de données avec le compte d'un analyste de données ou, en tant qu'analyste de données, en utilisant un ARN Lambda partagé d'un administrateur de données pour l'ajouter à votre compte. Lorsque des modifications de configuration sont apportées à un connecteur du compte d'origine, la configuration mise à jour est automatiquement appliquée aux instances partagées du connecteur dans les comptes d'autres utilisateurs.

Considérations et restrictions

- La fonction de requête fédérée entre comptes est disponible pour les connecteurs de données de métastores non Hive qui utilisent une source de données Lambda.
- La fonctionnalité n'est pas disponible pour le type de source de AWS Glue Data Catalog données. Pour plus d'informations sur l'accès entre comptes à AWS Glue Data Catalog s, consultezAccès entre comptes aux catalogues de données AWS Glue.
- Si la réponse de la fonction Lambda de votre connecteur dépasse la limite de taille de réponse Lambda de 6 Mo, Athena chiffre, regroupe et déverse automatiquement la réponse dans un compartiment Amazon S3 que vous configurez. L'entité qui exécute la requête Athena doit avoir

accès à l'emplacement du déversement pour qu'Athéna puisse lire les données déversées. Nous vous recommandons de définir une politique de cycle de vie Amazon S3 afin de supprimer les objets présents sur le lieu du déversement, car les données ne sont plus nécessaires une fois la requête terminée.

- L'utilisation de requêtes fédérées n' Régions AWS est pas prise en charge.

Autorisations nécessaires

- Pour que le compte administrateur de données A partage une fonction Lambda avec le compte d'analyste de données B, le compte B nécessite la fonction d'invocation Lambda et l'accès au compartiment de déversement. Par conséquent, le compte A devrait ajouter une politique basée sur les ressources à la fonction Lambda et l'accès principal à son compartiment de déversement dans Amazon S3.

  1. La politique suivante accorde des autorisations de fonction d'invocation Lambda au compte B sur une fonction Lambda dans le compte A.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CrossAccountInvocationStatement",
      "Effect": "Allow",
      "Principal": {
        "AWS": ["arn:aws:iam::account-B-id:user/username"]
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:aws-region:account-A-id:function:lambda-function-name"
    }
  ]
}
```

  2. La politique suivante permet d'accéder au compartiment de déversement au principal dans le compte B.

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": ["arn:aws:iam::account-B-id:user/username"]
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
             ],
            "Resource": [
                "arn:aws:s3:::spill-bucket",
                "arn:aws:s3:::spill-bucket/*"
            ]
        }
    ]
}
```

3. Si la fonction Lambda chiffre le bucket de spill avec une AWS KMS clé au lieu du chiffrement par défaut proposé par le SDK de fédération, la politique de AWS KMS clés du compte A doit accorder l'accès à l'utilisateur du compte B, comme dans l'exemple suivant.

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal":
    {
       "AWS": ["arn:aws:iam::account-B-id:user/username"]
    },
    "Action": [ "kms:Decrypt" ],
    "Resource": "*" // Resource policy that gets placed on the KMS key.
}
```

- Pour que le compte A partage son connecteur avec le compte B, le compte B doit créer un rôle appelé `AthenaCrossAccountCreate-account-A-id` que le compte A assume en appelant l'action AssumeRoleAPI AWS Security Token Service.

  La politique suivante, qui autorise l'action `CreateDataCatalog`, doit être créée dans le compte B et ajoutée au rôle `AthenaCrossAccountCreate-account-A-id` créé par le compte B pour le compte A.

```
{
 "Effect": "Allow",
 "Action": "athena:CreateDataCatalog",
 "Resource": "arn:aws:athena:*:account-B-id:datacatalog/*"
```

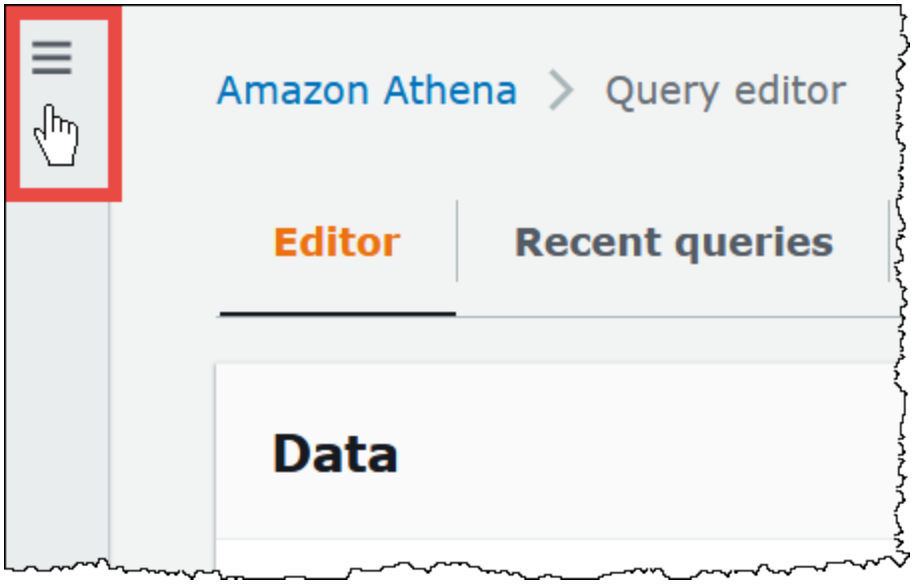
```
}
```

Partage d'une source de données dans le compte A avec le compte B

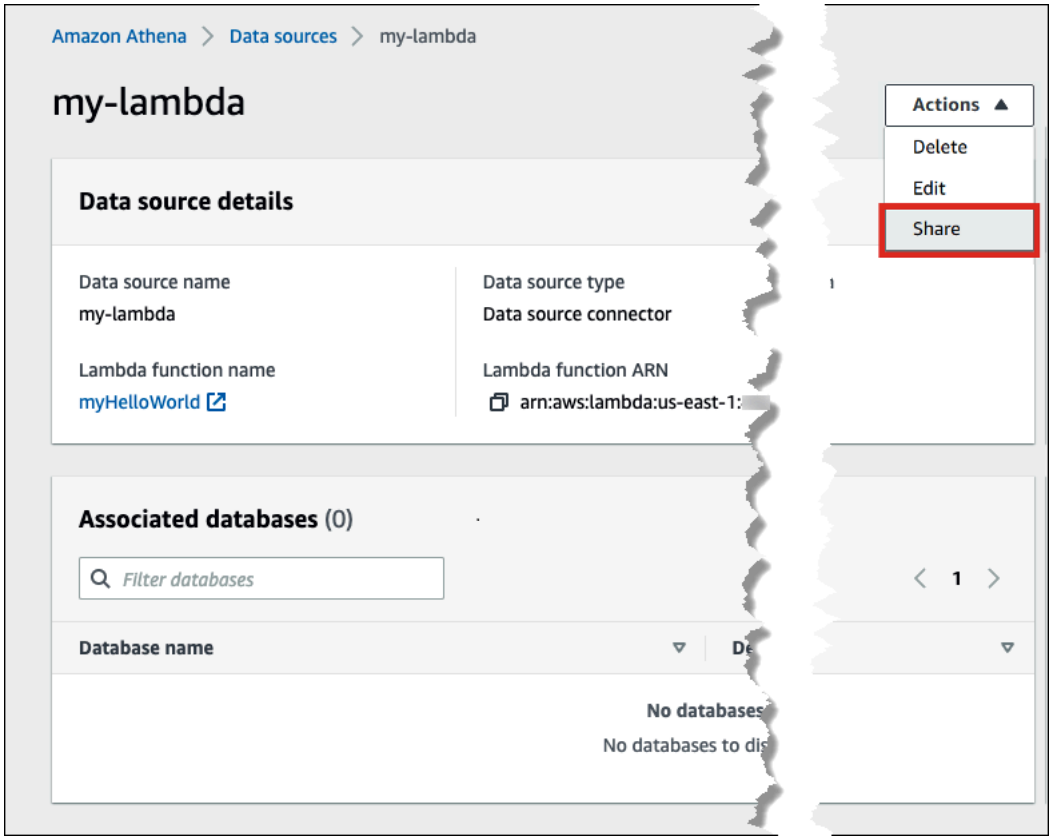
Une fois les autorisations en place, vous pouvez utiliser la page Data sources (Sources de données) dans la console Athena pour partager un connecteur de données dans votre compte (compte A) avec un autre compte (compte B). Le compte A conserve le contrôle total et la propriété du connecteur. Lorsque le compte A modifie la configuration du connecteur, la configuration mise à jour s'applique au connecteur partagé dans le compte B.

Pour partager une source de données Lambda du compte A avec le compte B

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

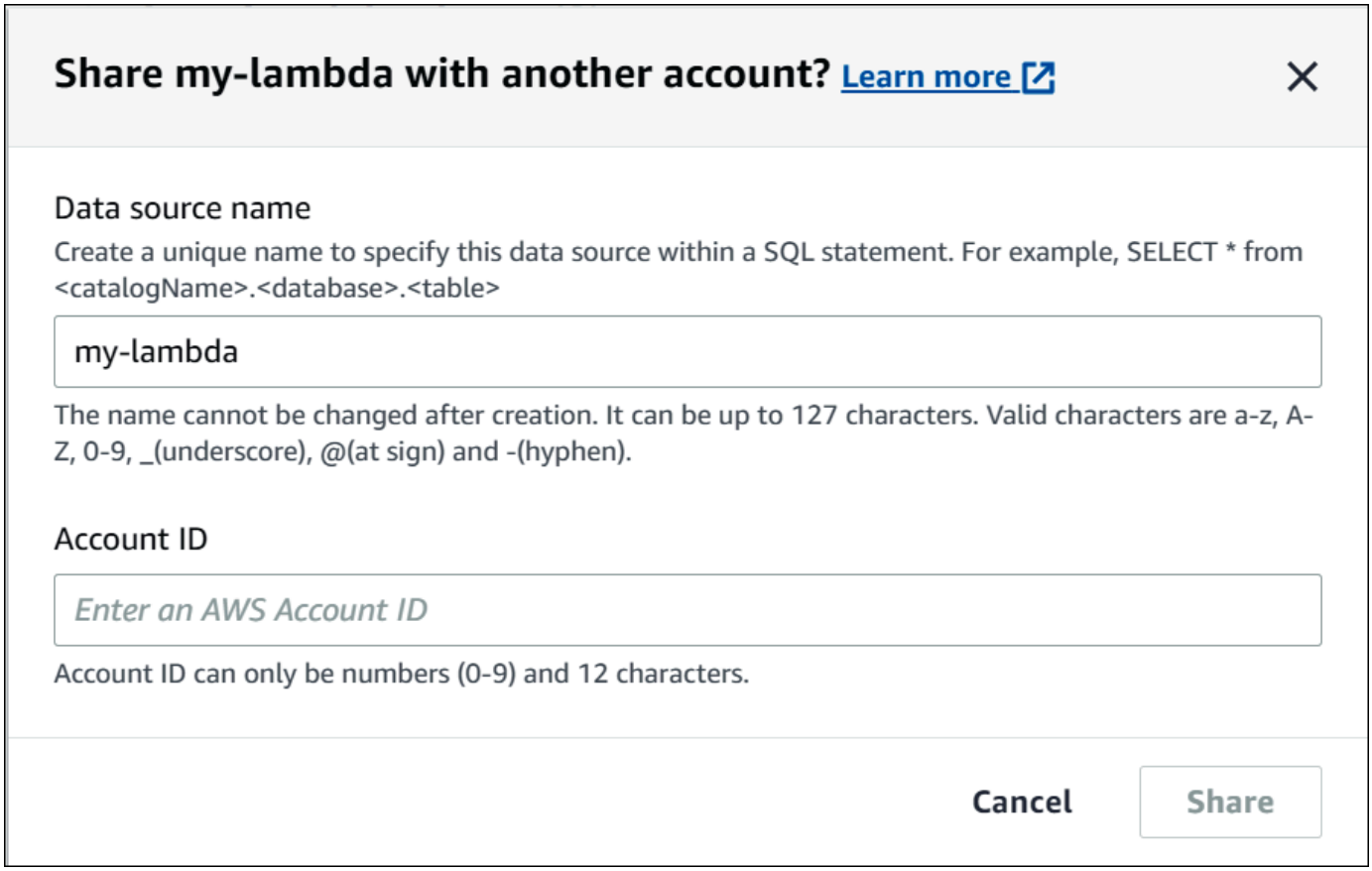


3. Choisissez Sources de données.
4. Sur la page Data sources (Sources de données), choisissez le lien du connecteur que vous souhaitez partager.
5. Sur la page des détails d'une source de données Lambda, choisissez l'option Share (Partager) dans le coin supérieur droit.



6. Dans Share (Partager) ***le nom Lambda*** avec un autre compte, saisissez les informations requises.

   - Pour Data source name (Nom de la source de données), saisissez le nom de la source de données copiée telle que vous souhaitez qu'elle apparaisse dans l'autre compte.
   - Pour Account ID (ID de compte), saisissez l'ID du compte avec lequel vous souhaitez partager votre source de données (dans ce cas, compte B).



7. Choisissez Partager. Le connecteur de données partagées que vous avez spécifié est créé dans le compte B. Les modifications de configuration apportées au connecteur dans le compte A s'appliquent au connecteur du compte B.

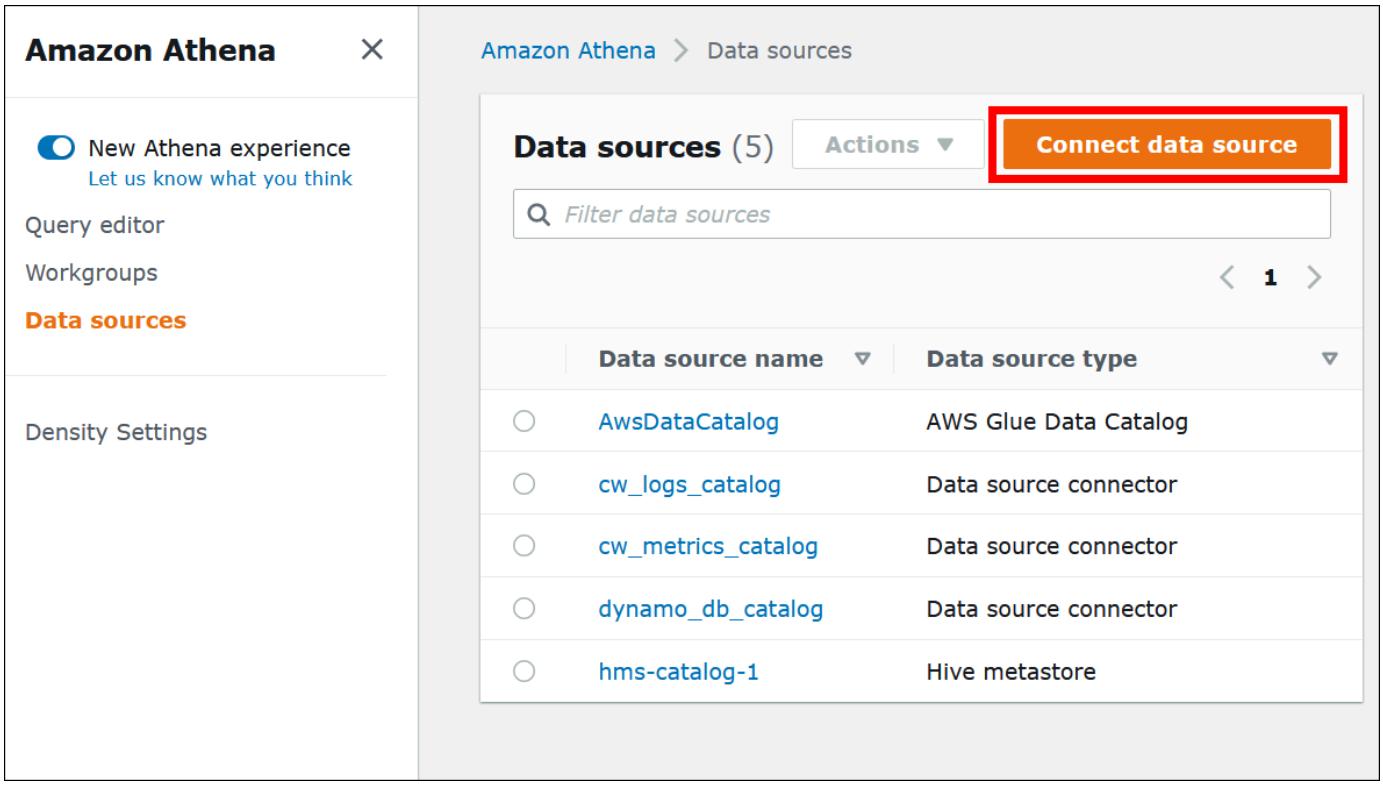
Ajout d'une source de données partagée du compte A au compte B

En tant qu'analyste de données, un administrateur de données peut vous attribuer l'ARN d'un connecteur à ajouter à votre compte. Vous pouvez utiliser la page des Sources de données de la console Athena pour ajouter l'ARN Lambda fourni par votre administrateur à votre compte.

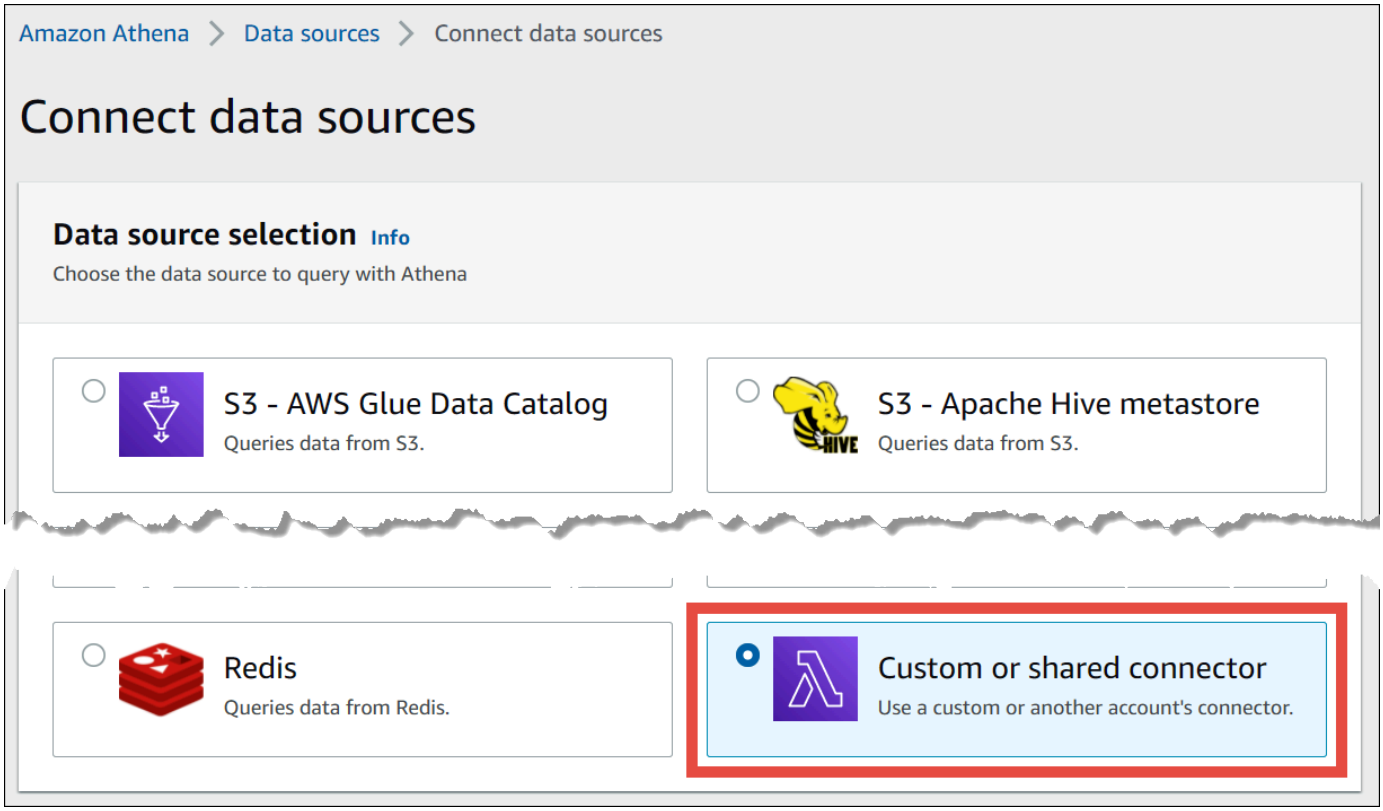
Pour ajouter l'ARN Lambda d'un connecteur de données partagées à votre compte

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si vous utilisez la nouvelle expérience de console et que le panneau de navigation n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Choisissez Sources de données.

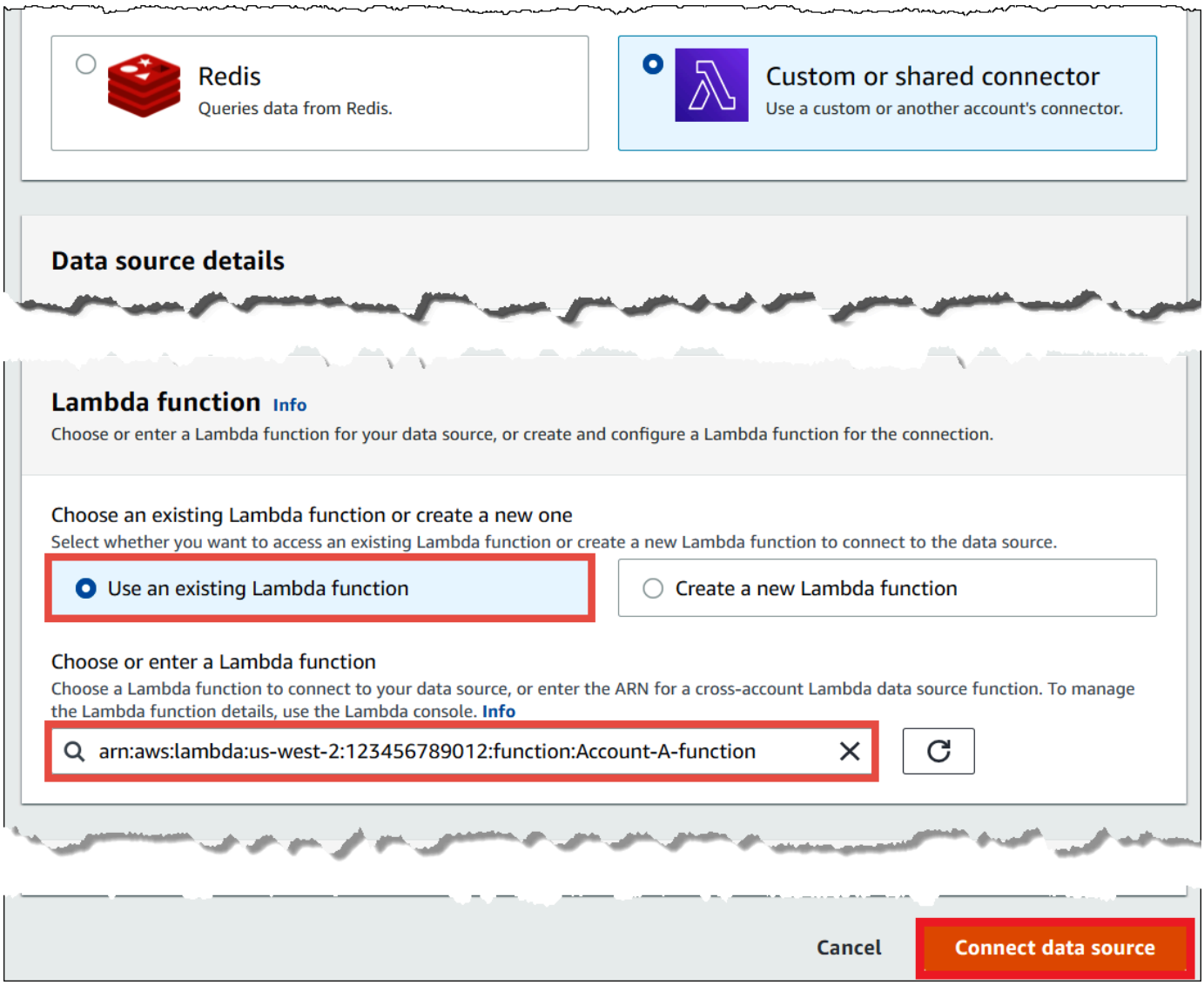
4. Sur la page Data sources (Sources de données), choisissez Connect data source (Connecter la source de données).



5. Choisissez Custom or shared connector (Connecteur personnalisé ou partagé).



6. Dans la section Lambda function (Fonction Lambda), assurez-vous que l'option Use an existing Lambda function (Utiliser une fonction Lambda existante) est sélectionnée.



7. Pour Choose or enter a Lambda function (Choisissez ou entrez une fonction Lambda), saisissez l'ARN Lambda du compte A.
8. Choisissez Connect data source (Connecter une source de données).

Résolution des problèmes

Si vous recevez un message d'erreur indiquant que le compte A ne dispose pas des autorisations nécessaires pour assumer un rôle dans le compte B, assurez-vous que le nom du rôle créé dans le compte B est correctement orthographié et que la politique appropriée est attachée.

Mise à jour d'un connecteur de source de données

Athena vous recommande de mettre régulièrement à jour les connecteurs de source de données que vous utilisez vers la dernière version afin de tirer parti des nouvelles fonctionnalités et améliorations. Pour commencer, vous devez trouver le dernier numéro de version.

Recherche de la dernière version d'Athena Query Federation

Le dernier numéro de version des connecteurs de source de données Athena correspond à la dernière version d'Athena Query Federation. Dans certains cas, les versions de GitHub peuvent être légèrement plus récentes que celles disponibles sur le AWS Serverless Application Repository (SAR).

Trouver le dernier numéro de version d'Athena Query Federation

1. Consultez l'URL GitHub https://github.com/awslabs/aws-athena-query-federation/releases/latest.
2. Notez le numéro de version dans l'en-tête de la page principale au format suivant :

   Version v *année*.*semaine_de_l_année*.*itération_de_la_semaine* d'Athena Query Federation

   Par exemple, le numéro de version de la Version v2023.8.3 d'Athena Query Federation est 2023.8.3.

Recherche et annotation des noms des ressources

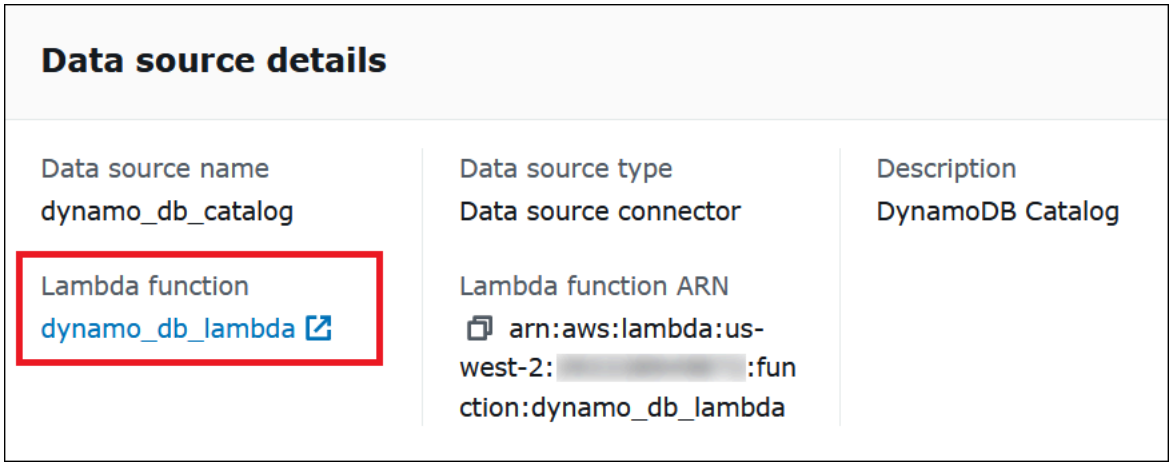
Pour préparer la mise à niveau, vous devez trouver et noter les informations suivantes :

1. Le nom de la fonction Lambda pour le connecteur.
2. Les variables d'environnement de la fonction Lambda.
3. Le nom de l'application Lambda, qui gère la fonction Lambda pour le connecteur.

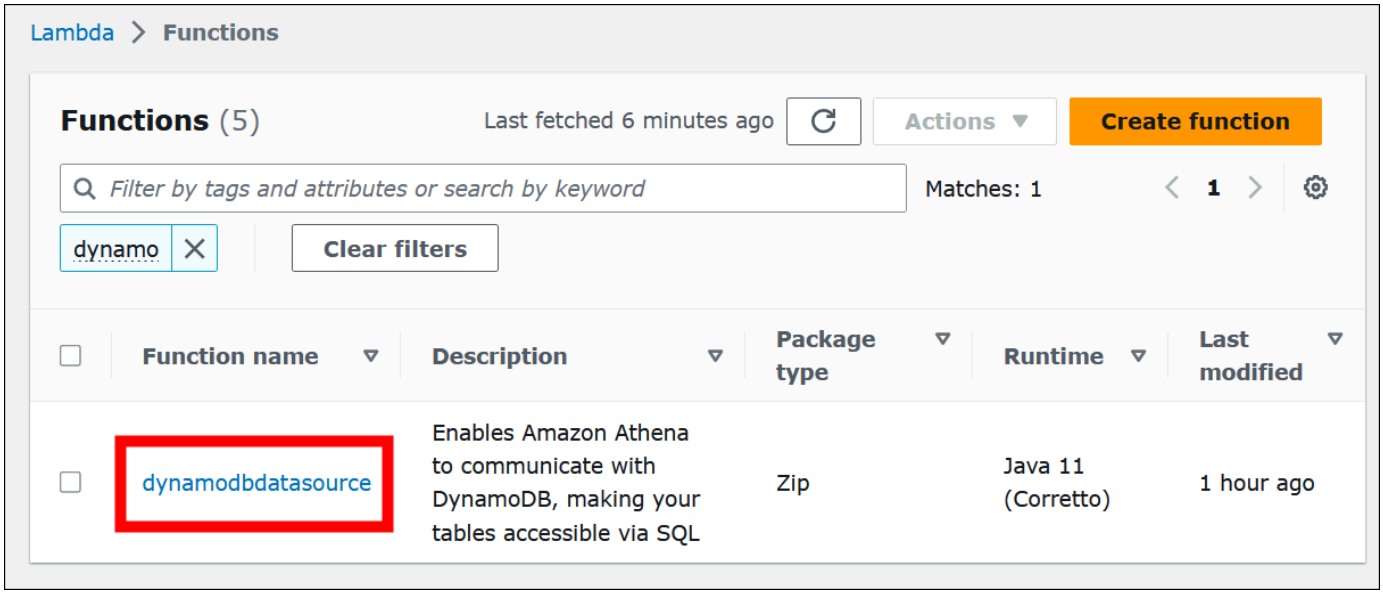
Pour trouver les noms des ressources dans la console Athena

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le panneau de navigation, choisissez Sources de données.
4. Dans la colonne Nom de la source de données, choisissez le lien vers la source de données de votre connecteur.
5. Dans la section Détails de la source de données, sous Fonction Lambda, choisissez le lien vers votre fonction Lambda.



6. Sur la page Fonctions, dans la colonne Nom de la fonction, notez le nom de la fonction de votre connecteur.



7. Choisissez le lien du nom de la fonction.
8. Dans la section Présentation de la fonction, choisissez l'onglet Configuration.
9. Dans le panneau de gauche, choisissez Variables d'environnement.
10. Dans la section Variables d'environnement, notez les clés et leurs valeurs correspondantes.
11. Faites défiler jusqu'en haut de la page.
12. Dans le message Cette fonction appartient à une application. Cliquez ici pour la gérer, sélectionnez le lien Cliquez ici.
13. Sur la page ***nom_de_votre_application*** serverlessrepo, notez le nom de votre application sans serverlessrepo. Par exemple, si le nom de l'application est serverlessrepo-DynamoDbTestApp, le nom de votre application est DynamoDbTestApp.
14. Restez sur la page de console Lambda pour votre application, puis suivez les étapes décrites dans la section Recherche de la version du connecteur que vous utilisez.

Recherche de la version du connecteur que vous utilisez

Pour trouver la version du connecteur que vous utilisez, procédez comme suit.

Pour trouver la version du connecteur que vous utilisez

1. Sur la page de console Lambda pour votre application Lambda, choisissez l'onglet Déploiements.
2. Dans l'onglet Déploiements, développez la section Modèle SAM.
3. Recherchez CodeUri.

4. Dans le champ Clé sous CodeUri, recherchez la chaîne suivante :

```
applications-connector_name-
versions-year.week_of_year.iteration_of_week/hash_number
```

   L'exemple suivant montre une chaîne pour le connecteur CloudWatch :

```
applications-AthenaCloudwatchConnector-versions-2021.42.1/15151159...
```

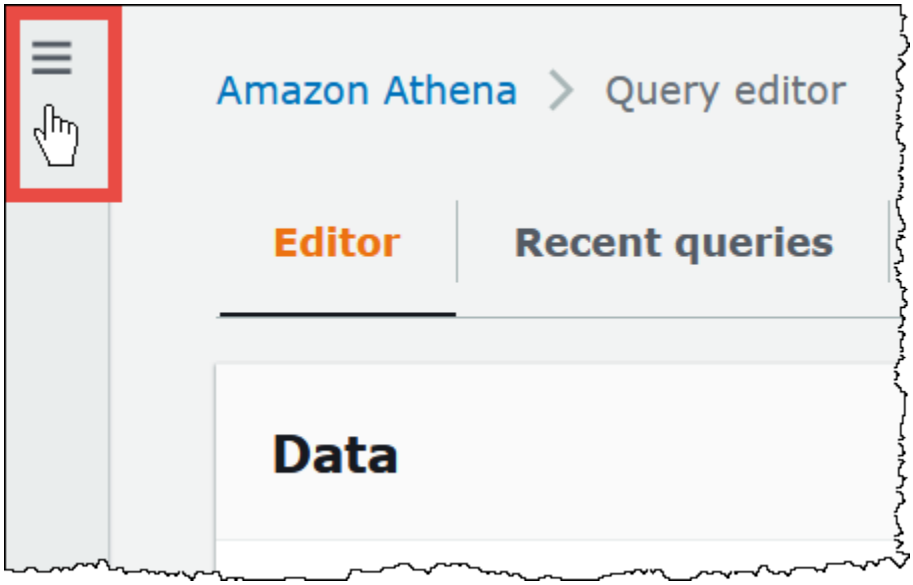
5. Enregistrez la valeur pour *année*.*semaine_de_l_année*.*itération_de_la_semaine* (par exemple, 2021.42.1). Il s'agit de la version correspondant à votre connecteur.

Déploiement de la nouvelle version de votre connecteur

Pour déployer une nouvelle version de votre connecteur, procédez comme suit.

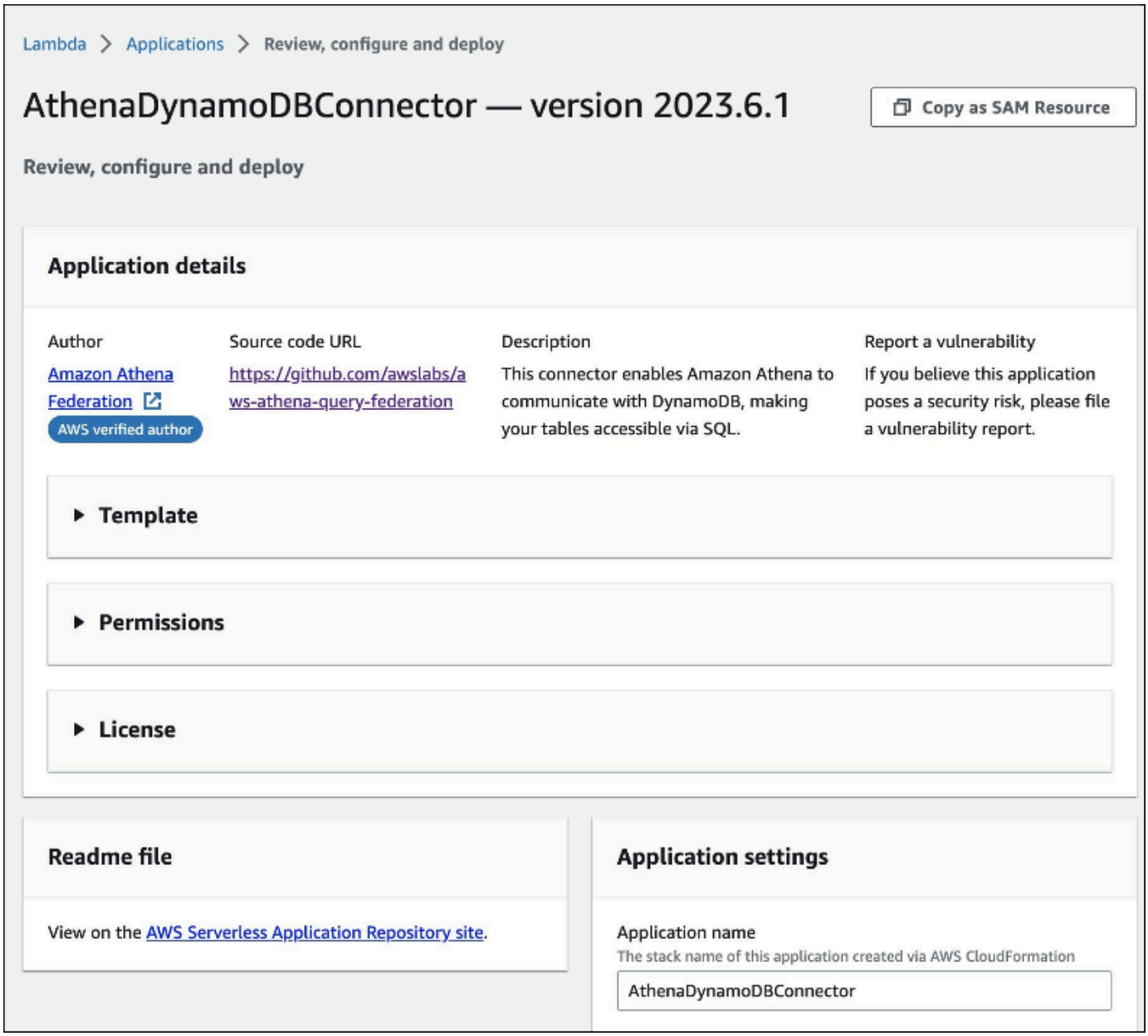
Pour déployer une nouvelle version de votre connecteur

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.



3. Dans le panneau de navigation, choisissez Sources de données.
4. À la page Data sources (Sources de données), choisissez Connect data source (Connecter la source de données).
5. Choisissez la source de données que vous souhaitez mettre à niveau, puis sélectionnez Suivant.

6. Dans la section Détails de la connexion, choisissez Créer une fonction Lambda. Cela ouvre la console Lambda dans laquelle vous pourrez déployer votre application mise à jour.



7. Comme vous ne créez pas réellement une nouvelle source de données, vous pouvez fermer l'onglet de la console Athena.
8. Sur la page de console Lambda pour le connecteur, effectuez les étapes suivantes :

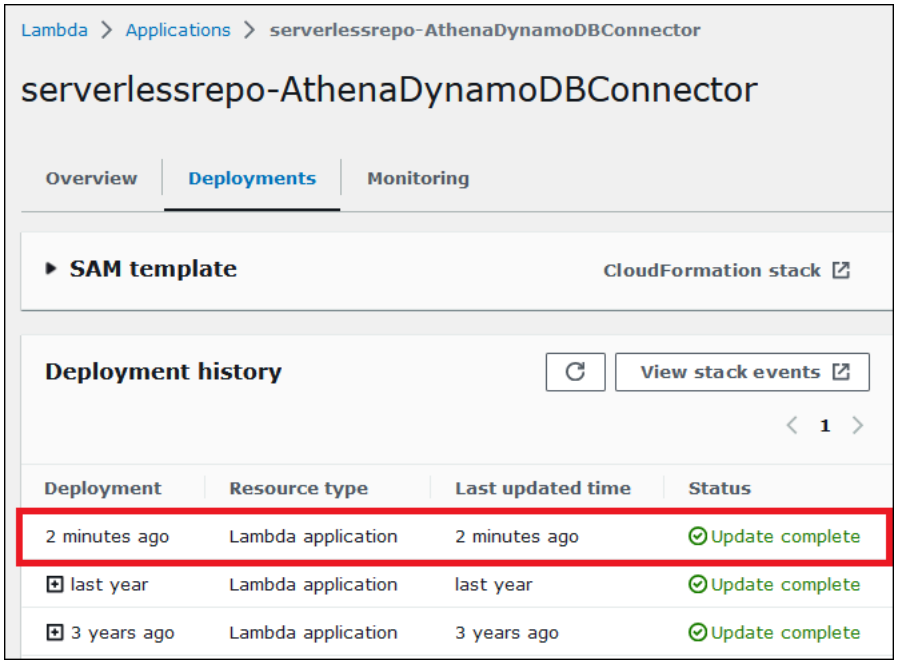
   a. Assurez-vous d'avoir supprimé le préfixe serverlessrepo- du nom de votre application, puis copiez le nom de l'application dans le champ Nom de l'application.

   b. Copiez le nom de votre fonction Lambda dans le champ AthenaCatalogName. Certains connecteurs appellent ce champ LambdaFunctionName.

c. Copiez les variables d'environnement que vous avez enregistrées dans les champs correspondants.

9. Sélectionnez l'option Je comprends que cette application crée des rôles IAM personnalisés, puis choisissez Déployer.
10. Pour vérifier que votre application a été mise à jour, sélectionnez l'onglet Déploiements.

    La section Historique de déploiement indique que votre mise à jour est terminée.



11. Pour confirmer le nouveau numéro de version, vous pouvez développer la section Modèle SAM comme auparavant, rechercher CodeUri et vérifier le numéro de version du connecteur dans le champ Clé.

Vous pouvez désormais utiliser votre connecteur mis à jour pour créer des requêtes fédérées Athena.

Exécution de requêtes fédérées

Après avoir configuré un ou plusieurs connecteurs de données et les avoir déployés sur votre compte, vous pouvez les utiliser dans vos requêtes Athena.

Interrogation d'une source de données unique

Les exemples de cette section supposent que vous avez configuré et déployé le Connecteur Amazon Athena CloudWatch sur votre compte. Utilisez la même approche pour effectuer des requêtes lorsque vous utilisez d'autres connecteurs.

Pour créer une requête Athena utilisant le connecteur CloudWatch

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans l'éditeur de requête Athena, créez une requête SQL utilisant la syntaxe suivante dans la clause `FROM`.

```
MyCloudwatchCatalog.database_name.table_name
```

Exemples

L'exemple suivant utilise le CloudWatch connecteur Athena pour se connecter à la `all_log_streams` vue du groupe `/var/ecommerce-engine/order-processor` CloudWatch Logs Log. La vue `all_log_streams` est une vue de tous les flux de journaux du groupe de journaux. L'exemple de requête limite le nombre de lignes renvoyées à 100.

```
SELECT *
FROM "MyCloudwatchCatalog"."/var/ecommerce-engine/order-processor".all_log_streams
LIMIT 100;
```

L'exemple suivant analyse les informations de la même vue que l'exemple précédent. L'exemple extrait l'ID de commande et le niveau du journal, puis filtre tout message ayant le niveau `INFO`.

```
SELECT
    log_stream as ec2_instance,
    Regexp_extract(message '.*orderId=(\d+) .*', 1) AS orderId,
    message AS order_processor_log,
    Regexp_extract(message, '(.*):.*', 1) AS log_level
FROM MyCloudwatchCatalog."/var/ecommerce-engine/order-processor".all_log_streams
WHERE Regexp_extract(message, '(.*):.*', 1) != 'INFO'
```

Interrogation de plusieurs sources de données

À titre d'exemple plus complexe, imaginez une entreprise de commerce électronique qui utilise les sources de données suivantes pour stocker les données relatives aux achats des clients :

- Amazon RDS for MySQL pour stocker les données du catalogue de produits
- Amazon DocumentDB pour stocker les données de compte client telles que les adresses e-mail et les adresses d'expédition

- Amazon DynamoDB pour stocker les données d'expédition et de suivi des commandes

Imaginez qu'un analyste de données pour cette application de commerce électronique apprenne que les délais de livraison dans certaines régions ont été affectés par les conditions météorologiques locales. L'analyste souhaite savoir combien de commandes sont retardées, où se trouvent les clients concernés et quels sont les produits les plus concernés. Au lieu d'étudier les sources d'informations séparément, l'analyste utilise Athena pour regrouper les données en une seule requête fédérée.

Example

```
SELECT
     t2.product_name AS product,
     t2.product_category AS category,
     t3.customer_region AS region,
     count(t1.order_id) AS impacted_orders
FROM my_dynamodb.default.orders t1
JOIN my_mysql.products.catalog t2 ON t1.product_id = t2.product_id
JOIN my_documentdb.default.customers t3 ON t1.customer_id = t3.customer_id
WHERE
     t1.order_status = 'PENDING'
     AND t1.order_date between '2022-01-01' AND '2022-01-05'
GROUP BY 1, 2, 3
ORDER BY 4 DESC
```

Interrogation des vues fédérées

Lorsque vous interrogez des sources fédérées, vous pouvez utiliser des vues afin d'obscurcir les sources de données sous-jacentes ou de masquer les jointures complexes pour les autres analystes qui interrogent les données.

Considérations et restrictions

- Les vues fédérées nécessitent la version 3 du moteur Athena.
- Les vues fédérées sont stockées dans la source de données sous-jacente AWS Glue, et non dans celle-ci.
- Les vues créées avec des catalogues fédérés doivent utiliser une syntaxe de nom complète, comme dans l'exemple suivant :

```
"ddbcatalog"."default"."customers"
```

- Les utilisateurs qui exécutent des requêtes sur des sources fédérées doivent être autorisés à interroger les sources fédérées.
- L'autorisation `athena:GetDataCatalog` est requise pour les vues fédérées. Pour plus d'informations, consultez Exemple de politiques d'autorisation IAM pour autoriser la requête fédérée Athena.

Exemples

L'exemple suivant crée une vue appelée `customers` sur les données stockées dans une source de données fédérée.

Example

```
CREATE VIEW customers AS
SELECT *
FROM my_federated_source.default.table
```

L'exemple de requête suivant montre une requête qui fait référence à la vue `customers` plutôt qu'à la source de données fédérée sous-jacente.

Example

```
SELECT id, SUM(order_amount)
FROM customers
GROUP by 1
ORDER by 2 DESC
LIMIT 50
```

L'exemple suivant crée une vue appelée `order_summary` qui combine les données d'une source de données fédérée et celles d'une source de données Amazon S3. À partir de la source fédérée, qui a déjà été créée dans Athena, la vue utilise les tables `person` et `profile`. Dans Amazon S3, la vue utilise les tables `purchase` et `payment`. Pour faire référence à Amazon S3, l'instruction utilise le mot-clé `awsdatacatalog`. Notez que la source de données fédérée utilise la syntaxe de nom complète *`federated_source_name`*.*`federated_source_database`*.*`federated_source_table`*.

Example

```
CREATE VIEW default.order_summary AS
```

```
SELECT *
FROM federated_source_name.federated_source_database."person" p
    JOIN federated_source_name.federated_source_database."profile" pr ON pr.id = p.id
    JOIN awsdatacatalog.default.purchase i ON p.id = i.id
    JOIN awsdatacatalog.default.payment pay ON pay.id = p.id
```

Ressources supplémentaires

- Pour un exemple de vue fédérée découplée de sa source d'origine et disponible pour une analyse à la demande dans un modèle d'utilisateurs multiples, consultez la section Étendre votre maillage de données avec Amazon Athena et les vues fédérées sur le blog AWS Big Data.
- Pour de plus amples informations sur le fonctionnement des vues dans Athena, consultez Utilisation des vues.

Exécution de requêtes directes fédérées

Dans Athena, vous pouvez exécuter des requêtes sur des sources de données fédérées en utilisant le langage de requête de la source de données elle-même et transférer la requête complète vers la source de données pour exécution. Ces requêtes sont appelées requêtes passthrough. Pour exécuter des requêtes directes, vous devez utiliser une fonction de table dans votre requête Athena. Vous incluez la requête directe à exécuter sur la source de données dans l'un des arguments de la fonction de table. Les requêtes passthrough renvoient une table que vous pouvez analyser à l'aide d'Athena SQL.

Connecteurs pris en charge

Les connecteurs de source de données Athena suivants prennent en charge les requêtes directes.

- Stockage Azure Data Lake
- Azure Synapse
- Cloudera Hive
- Cloudera Impala
- CloudWatch
- Db2
- Série Db2 i
- DocumentDB
- DynamoDB

- HBase
- Google BigQuery
- Hortonworks
- MySQL
- Neptune
- OpenSearch
- Oracle
- PostgreSQL
- Redshift
- SAP HANA
- Snowflake
- SQL Server
- Teradata
- Timestream
- Vertica

Considérations et restrictions

Lorsque vous utilisez des requêtes directes dans Athena, tenez compte des points suivants :

- Le transfert de requêtes n'est pris en charge que pour les instructions `SELECT` Athena ou les opérations de lecture.
- Les requêtes passthrough doivent être exécutées dans le contexte du catalogue de la requête externe (c'est-à-dire de la requête qui appelle la fonction de table).
- Les performances des requêtes peuvent varier en fonction de la configuration de la source de données.
- Les requêtes directes ne sont pas prises en charge pour les vues.

Syntaxe

La syntaxe générale de transmission des requêtes Athena est la suivante.

```
SELECT * FROM TABLE(system.function_name(arg1 => 'arg1Value'[, arg2 =>
 'arg2Value', ...]))
```

Pour la plupart des sources de données, le premier et unique argument est `query` suivi de l'opérateur flèche => et de la chaîne de requête.

```
SELECT * FROM TABLE(system.query(query => 'query string'))
```

Pour des raisons de simplicité, vous pouvez omettre l'argument nommé facultatif `query` et l'opérateur => flèche.

```
SELECT * FROM TABLE(system.query('query string'))
```

Si la source de données nécessite plus que la chaîne de requête, utilisez des arguments nommés dans l'ordre attendu par la source de données. Par exemple, l'expression *`arg1`* `=>` '*`arg1Value`*' contient le premier argument et sa valeur. Le nom *`arg1`* est spécifique à la source de données et peut varier d'un connecteur à l'autre.

```
SELECT * FROM TABLE(
        system.query(
            arg1 => 'arg1Value',
            arg2 => 'arg2Value',
            arg3 => 'arg3Value'
        ));
```

Pour plus d'informations sur la syntaxe exacte à utiliser avec un connecteur particulier, consultez la page du connecteur individuel.

Utilisation des guillemets

Les valeurs des arguments, y compris la chaîne de requête que vous transmettez, doivent être placées entre guillemets simples, comme dans l'exemple suivant.

```
SELECT * FROM TABLE(system.query(query => 'SELECT * FROM testdb.persons LIMIT 10'))
```

Lorsque la chaîne de requête est entourée de guillemets, la requête échoue. La requête suivante échoue avec le message d'erreur COLUMN_NOT_FOUND : ligne 1:43 : La colonne 'select * from testdb.persons limit 10' ne peut pas être résolue.

```
SELECT * FROM TABLE(system.query(query => "SELECT * FROM testdb.persons LIMIT 10"))
```

Pour éviter un guillemet unique, ajoutez un seul guillemet à l'original (par exemple, `terry's_group` à`terry''s_group`).

Exemples

L'exemple de requête suivant permet de transférer une requête vers une source de données. La requête sélectionne toutes les colonnes de la `customer` table, limitant les résultats à 10.

```
SELECT * FROM TABLE(
        system.query(
            query => 'SELECT * FROM customer LIMIT 10;'
        ))
```

L'instruction suivante exécute la même requête, mais élimine l'argument nommé facultatif `query` et l'opérateur flèche=>.

```
SELECT * FROM TABLE(
        system.query(
            'SELECT * FROM customer LIMIT 10;'
        ))
```

Athena et qualificateurs de noms de tables fédérées

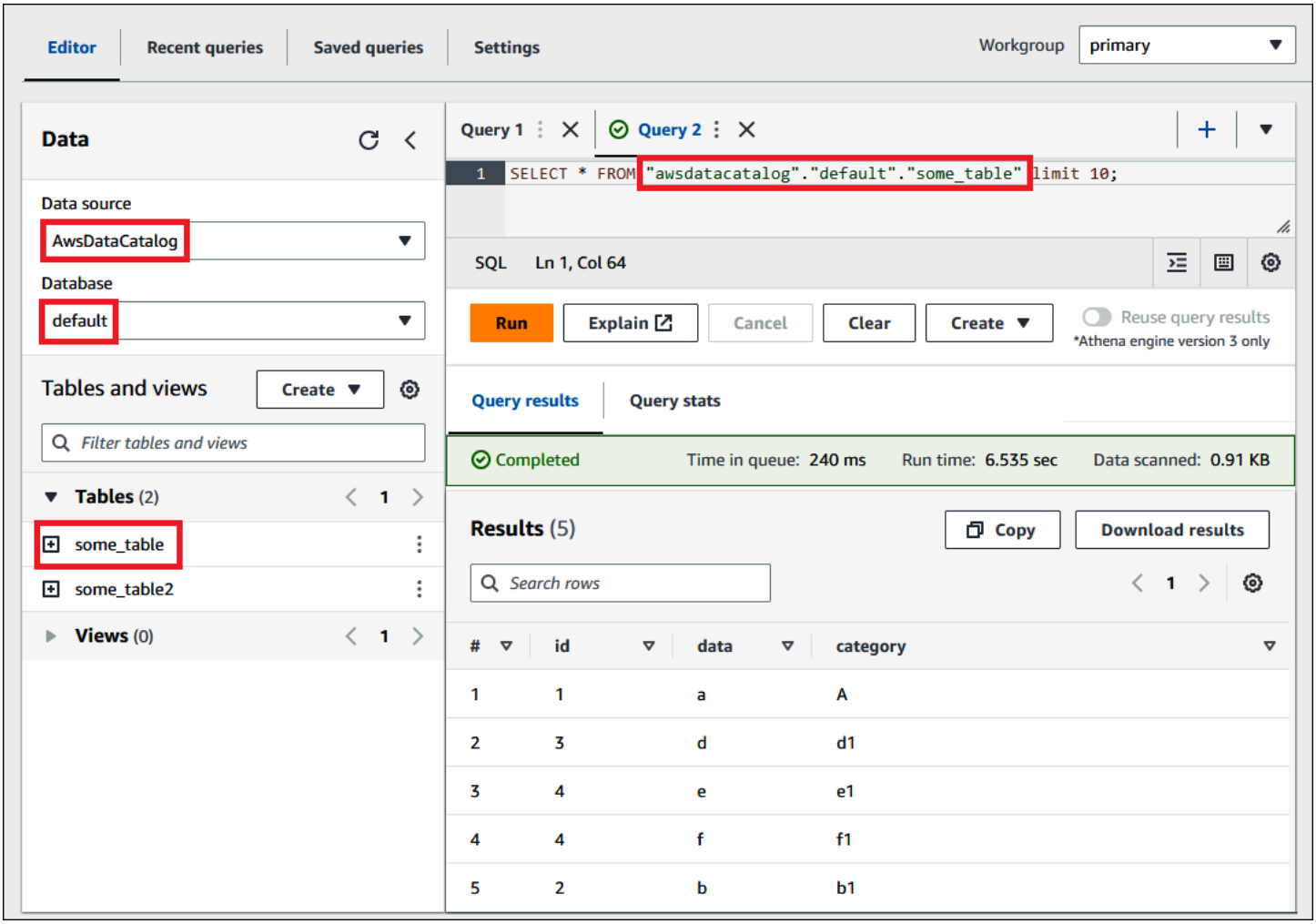
Athena utilise les termes suivants pour désigner les hiérarchies d'objets de données :

- Source de données : un groupe de bases de données
- Base de données : un groupe de tables
- Table : des données organisées sous la forme d'un groupe de lignes ou de colonnes

Parfois, ces objets sont également désignés par des noms alternatifs mais équivalents, tels que les suivants :

- Une source de données est parfois appelée catalogue.
- Une base de données est parfois appelée schéma.

L'exemple de requête suivant dans la console Athena utilise la source de données `awsdatacatalog`, la base de données `default` et la table `some_table`.



Termes utilisés dans les sources de données fédérées

Lorsque vous interrogez des sources de données fédérées, notez que la source de données sous-jacente n'utilise peut-être pas la même terminologie qu'Athena. Gardez cette distinction à l'esprit lorsque vous rédigez vos requêtes fédérées. Les sections suivantes décrivent comment les termes des objets de données dans Athena correspondent à ceux des sources de données fédérées.

Amazon Redshift

Une base de données Amazon Redshift est un groupe de schémas Redshift contenant un groupe de tables Redshift.

Athena	Redshift
Source de données Redshift	Une fonction Lambda du connecteur Redshift configurée pour pointer vers une `database` Redshift.
`data_source.database.table`	`database.schema.table`

Exemple de requête

```
SELECT * FROM
Athena_Redshift_connector_data_source.Redshift_schema_name.Redshift_table_name
```

Pour en savoir plus sur ce connecteur, consultez Connecteur Amazon Athena pour Redshift.

Cloudera Hive

Un serveur ou un cluster Cloudera Hive est un groupe de bases de données Cloudera Hive contenant un groupe de tables Cloudera Hive.

Athena	Hive
Source de données Cloudera Hive	Fonction Lambda du connecteur Cloudera Hive configurée pour pointer vers un `server` Cloudera Hive.
`data_source.database.table`	`server.database.table`

Exemple de requête

```
SELECT * FROM
Athena_Cloudera_Hive_connector_data_source.Cloudera_Hive_database_name.Cloudera_Hive_table_name
```

Pour en savoir plus sur ce connecteur, consultez Connecteur Amazon Athena pour Cloudera Hive.

Cloudera Impala

Un serveur ou un cluster Impala est un groupe de bases de données Impala contenant un groupe de tables Impala.

Athena	Impala
Source de données Impala	Fonction Lambda du connecteur Impala configurée pour pointer vers un `server` Impala.
`data_source.database.table`	`server.database.table`

Exemple de requête

```
SELECT * FROM
Athena_Impala_connector_data_source.Impala_database_name.Impala_table_name
```

Pour en savoir plus sur ce connecteur, consultez Connecteur Amazon Athena pour Cloudera Impala.

MySQL

Un serveur MySQL est un groupe de bases de données MySQL contenant un groupe de tables MySQL.

Athena	MySQL
Source de données MySQL	Fonction Lambda du connecteur MySQL configurée pour pointer vers un `server` MySQL.
`data_source.database.table`	`server.database.table`

Exemple de requête

```
SELECT * FROM
Athena_MySQL_connector_data source.MySQL_database_name.MySQL_table_name
```

Pour en savoir plus sur ce connecteur, consultez Connecteur Amazon Athena pour MySQL.

Oracle

Un serveur (ou une base de données) Oracle est un groupe de schémas Oracle contenant un groupe de tables Oracle.

Athena	Oracle
Source de données Oracle	Fonction Lambda du connecteur Oracle configurée pour pointer vers un `server` Oracle.
`data_source.database.table`	`server.schema.table`

Exemple de requête

```
SELECT * FROM
Athena_Oracle_connector_data_source.Oracle_schema_name.Oracle_table_name
```

Pour en savoir plus sur ce connecteur, consultez Connecteur Amazon Athena pour Oracle.

Postgres

Un serveur (ou cluster) Postgres est un groupe de bases de données Postgres. Une base de données Postgres est un groupe de schémas Postgres contenant un groupe de tables Postgres.

Athena	Postgres
Source de données Postgres	Fonction Lambda du connecteur Postgres configurée pour pointer vers un `server` et une `database` Postgres.
`data_source.database.table`	`server.database.schema.table`

Exemple de requête

```
SELECT * FROM
```

```
Athena_Postgres_connector_data_source.Postgres_schema_name.Postgres_table_name
```

Pour en savoir plus sur ce connecteur, consultez Connecteur Amazon Athena pour PostgreSQL.

Développement d'un connecteur de source de données à l'aide du kit SDK Athena Query Federation

Pour écrire vos propres connecteurs de source de données, vous pouvez utiliser le kit Athena Query Federation SDK. Le kit Athena Query Federation SDK définit un ensemble d'interfaces et de protocoles filaire que vous pouvez utiliser pour permettre à Athena de déléguer des parties de son plan d'exécution de requêtes au code que vous écrivez et déployez. Le kit SDK comprend une suite de connecteurs et un exemple de connecteur.

Vous pouvez également personnaliser les connecteurs prédéfinis d'Amazon Athena pour votre propre usage. Vous pouvez modifier une copie du code source depuis GitHub puis utiliser l'outil de publication Connector pour créer votre propre AWS Serverless Application Repository package. Une fois que vous avez déployé votre connecteur de cette manière, vous pouvez l'utiliser dans vos requêtes Athena.

Pour plus d'informations sur le téléchargement du SDK et pour obtenir des instructions détaillées sur l'écriture de votre propre connecteur, consultez Example Athena connector on. GitHub

Connecteurs de source de données Athena pour Apache Spark

Certains connecteurs de source de données Athena sont disponibles sous forme de connecteurs Spark DSV2. Les noms des connecteurs Spark DSV2 ont un suffixe `-dsv2` (par exemple,`athena-dynamodb-dsv2`).

Voici ci-dessous les connecteurs DSV2 actuellement disponibles, le nom de leur classe `.format()` Spark et les liens vers leur documentation Amazon Athena Federated Query correspondante :

Connecteur DSV2	Nom de classe .format () Spark	Documentation
athena-cloudwatch-dsv2	`com.amazonaws.athena.connectors.dsv2.cloudwatch.CloudwatchTableProvider`	CloudWatch

Connecteur DSV2	Nom de classe .format () Spark	Documentation
athena-cloudwatch-metrics-dsv2	`com.amazonaws.athena.connectors.dsv2.cloudwatch.metrics.CloudwatchMetricsTableProvider`	CloudWatch métriques
athena-aws-cmdb-dsv2	`com.amazonaws.athena.connectors.dsv2.aws.cmdb.AwsCmdbTableProvider`	CMDB
athena-dynamodb-dsv2	`com.amazonaws.athena.connectors.dsv2.dynamodb.DDBTableProvider`	DynamoDB

Pour télécharger **`.jar`** des fichiers pour les connecteurs DSV2, rendez-vous sur la page GitHub DSV2 d'Amazon Athena Query Federation et consultez la section Releases, Release, Assets. <version>

Spécification du fichier jar dans Spark

Pour utiliser les connecteurs Athena DSV2 avec Spark, vous devez transmettre le fichier `.jar` du connecteur à l'environnement Spark que vous utilisez. Les sections suivantes décrivent les cas spécifiques.

Athena pour Spark

Pour plus d'informations sur l'ajout de fichiers `.jar` personnalisés et d'une configuration personnalisée dans Amazon Athena pour Apache Spark, consultez. Ajout de fichiers JAR et de la configuration personnalisée de Spark

General Spark

Pour transmettre le fichier `.jar` du connecteur à Spark, utilisez la commande `spark-submit` et spécifiez le fichier `.jar` dans l'option `--jars`, comme dans l'exemple suivant :

```
spark-submit \
  --deploy-mode cluster \
  --jars https://github.com/awslabs/aws-athena-query-federation-dsv2/releases/
download/some_version/athena-dynamodb-dsv2-some_version.jar
```

Amazon EMR Spark

Pour exécuter une commande `spark-submit` avec le paramètre `--jars` sur Amazon EMR, vous devez ajouter une étape à votre cluster Amazon EMR Spark. Pour en savoir plus sur l'utilisation de `spark-submit` sur Amazon EMR, consultez Ajouter une étape Spark dans le Guide de version d'Amazon EMR.

AWS Glue ETL Spark

Pour l' AWS Glue ETL, vous pouvez transmettre l'URL GitHub .com du `.jar` fichier à l'`--extra-jars`argument de la `aws glue start-job-run` commande. La AWS Glue documentation décrit le `--extra-jars` paramètre comme empruntant un chemin Amazon S3, mais il peut également prendre une URL HTTPS. Pour plus d'informations, consultez la référence du paramètre de tâche dans le Guide du développeur AWS Glue .

Interrogation du connecteur sur Spark

Pour envoyer l'équivalent de votre requête fédérée Athena existante sur Apache Spark, utilisez la fonction `spark.sql()`. Supposons par exemple que vous ayez la requête Athena que vous souhaitez utiliser sur Apache Spark.

```
SELECT somecola, somecolb, somecolc
FROM ddb_datasource.some_schema_or_glue_database.some_ddb_or_glue_table
WHERE somecola > 1
```

Pour exécuter la même requête sur Spark à l'aide du connecteur Amazon Athena DynamoDB DSV2, utilisez le code suivant :

```
dynamoDf = (spark.read
    .option("athena.connectors.schema", "some_schema_or_glue_database")
    .option("athena.connectors.table", "some_ddb_or_glue_table")
    .format("com.amazonaws.athena.connectors.dsv2.dynamodb.DDBTableProvider")
    .load())
```

```
dynamoDf.createOrReplaceTempView("ddb_spark_table")

spark.sql('''
SELECT somecola, somecolb, somecolc
FROM ddb_spark_table
WHERE somecola > 1
''')
```

Spécification des paramètres

Les versions DSV2 des connecteurs de source de données Athena utilisent les mêmes paramètres que les connecteurs de sources de données Athena correspondants. Pour obtenir des informations sur les paramètres, reportez-vous à la documentation du connecteur de source de données Athena correspondant.

Dans votre PySpark code, utilisez la syntaxe suivante pour configurer vos paramètres.

```
spark.read.option("athena.connectors.conf.parameter", "value")
```

Par exemple, le code suivant définit le paramètre `disable_projection_and_casing` du connecteur Amazon Athena DynamoDB sur `always`.

```
dynamoDf = (spark.read
    .option("athena.connectors.schema", "some_schema_or_glue_database")
    .option("athena.connectors.table", "some_ddb_or_glue_table")
    .option("athena.connectors.conf.disable_projection_and_casing", "always")
    .format("com.amazonaws.athena.connectors.dsv2.dynamodb.DDBTableProvider")
    .load())
```

Politiques IAM pour l'accès aux catalogues de données

Pour contrôler l'accès aux catalogues de données, utilisez les autorisations IAM au niveau des ressources ou les politiques IAM basées sur l'identité.

La procédure suivante est spécifique à Athena.

Pour des informations spécifiques à IAM, consultez les liens répertoriés à la fin de cette section. Pour de plus amples informations sur les stratégies de catalogue de données JSON, veuillez consulter Exemple de politiques de catalogue de données.

Utilisation de l'éditeur visuel dans la console IAM pour créer une politique de catalogue de données

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation à gauche, choisissez politiques, puis Créer une politique.
3. Dans l'onglet Visual editor (Éditeur visuel), sélectionnez Choose a service (Choisir un service). Choisissez ensuite Athena pour l'ajouter à la politique.
4. Choisissez Sélectionner des actions, puis choisissez les actions à ajouter à la politique. L'éditeur visuel affiche les actions disponibles dans Athena. Pour plus d'informations, consultez la rubrique Actions, ressources et clés de condition pour Amazon Athena dans la section Référence de l'autorisation de service.
5. Choisissez Add actions (Ajouter des actions) pour entrer une action spécifique ou utilisez des caractères génériques (*) pour spécifier plusieurs actions.

   Par défaut, la politique que vous créez autorise les actions que vous choisissez. Si vous avez choisi une ou plusieurs actions qui prennent en charge les autorisations au niveau des ressources pour la ressource `datacatalog` dans Athena, l'éditeur visuel affiche la ressource `datacatalog`.
6. Choisissez Ressources pour spécifier les catalogues de données spécifiques à votre stratégie. Par exemple, des stratégies de catalogue de données JSON, veuillez consulter Exemple de politiques de catalogue de données.
7. Spécifiez la ressource du `datacatalog` comme suit :

   ```
   arn:aws:athena:<region>:<user-account>:datacatalog/<datacatalog-name>
   ```
8. Choisissez Review policy (Examiner une politique), puis saisissez un Name (Nom) et une Description (facultatif) pour la politique que vous êtes en train de créer. Passez en revue le résumé de politique afin de vous assurer que les autorisations nécessaires vous ont été accordées.
9. Choisissez Create policy (Créer une politique) pour enregistrer votre nouvelle politique.
10. Attachez cette stratégie basée sur une identité à un utilisateur, un groupe ou un rôle et spécifiez les ressources `datacatalog` auxquelles ils peuvent accéder.

Pour plus d'informations, consultez les rubriques suivantes dans la section Référence de l'autorisation de service et le Guide de l'utilisateur IAM :

- Actions, ressources et clés de condition pour Amazon Athena
- Création de politiques avec l'éditeur visuel
- Ajout et suppression de politiques IAM
- Contrôle de l'accès aux ressources

Par exemple, des stratégies de catalogue de données JSON, veuillez consulter Exemple de politiques de catalogue de données.

Pour plus d'informations sur AWS Glue les autorisations et les autorisations d' AWS Glue explorateur, consultez la section Configuration des autorisations IAM AWS Glue et des conditions requises pour les robots d'exploration dans le guide du développeur.AWS Glue

Pour obtenir une liste complète d'actions Amazon Athena, consultez les noms d'action API dans la rubrique Référence d'API Amazon Athena.

Exemple de politiques de catalogue de données

Cette section inclut des exemples de stratégies que vous pouvez utiliser pour activer plusieurs actions sur des catalogues de données.

Le catalogue de données est une ressource IAM gérée par Athena. Par conséquent, si votre stratégie de catalogue de données utilise des actions qui prennent `datacatalog` en entrée, vous devez spécifier l'ARN du catalogue de données comme suit :

```
"Resource": [arn:aws:athena:<region>:<user-account>:datacatalog/<datacatalog-name>]
```

Le `<datacatalog-name>` est le nom de votre catalogue de données. Par exemple, pour un catalogue de données nommé `test_datacatalog`, spécifiez-le en tant que ressource comme suit :

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:datacatalog/test_datacatalog"]
```

Pour obtenir une liste complète d'actions Amazon Athena, consultez les noms d'action API dans la rubrique Référence d'API Amazon Athena. Pour plus d'informations sur les politiques IAM, consultez la rubrique Création de politiques avec l'éditeur visuel du Guide de l'utilisateur IAM. Pour plus d'informations sur la création de politiques IAM pour les groupes de travail, voir Politiques IAM pour l'accès aux catalogues de données.

- Example Policy for Full Access to All Data Catalogs

- Example Policy for Full Access to a Specified Data Catalog
- Example Policy for Querying a Specified Data Catalog
- Example Policy for Management Operations on a Specified Data Catalog
- Example Policy for Listing Data Catalogs
- Example Policy for Metadata Operations on Data Catalogs

Example Exemple de politique pour un accès complet à tous les catalogues de données

La stratégie suivante permet un accès complet à toutes les ressources de catalogues de données susceptibles d'exister dans le compte. Nous vous recommandons d'utiliser cette stratégie pour ces utilisateurs dans votre compte qui doivent administrer et gérer les catalogues de données pour tous les autres utilisateurs.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "athena:*"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Example Exemple de politique pour un accès complet à un catalogue de données spécifié

La stratégie suivante autorise un accès complet à une seule ressource de catalogue de données spécifique nommée `datacatalogA`. Vous pouvez utiliser cette stratégie pour les utilisateurs ayant un contrôle total sur un catalogue de données en particulier.

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

```
            "athena:ListDataCatalogs",
            "athena:ListWorkGroups",
            "athena:GetDatabase",
            "athena:ListDatabases",
            "athena:ListTableMetadata",
            "athena:GetTableMetadata"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "athena:StartQueryExecution",
            "athena:GetQueryResults",
            "athena:DeleteNamedQuery",
            "athena:GetNamedQuery",
            "athena:ListQueryExecutions",
            "athena:StopQueryExecution",
            "athena:GetQueryResultsStream",
            "athena:ListNamedQueries",
            "athena:CreateNamedQuery",
            "athena:GetQueryExecution",
            "athena:BatchGetNamedQuery",
            "athena:BatchGetQueryExecution",
            "athena:DeleteWorkGroup",
            "athena:UpdateWorkGroup",
            "athena:GetWorkGroup",
            "athena:CreateWorkGroup"
         ],
         "Resource":[
            "arn:aws:athena:us-east-1:123456789012:workgroup/*"
         ]
      },
      {
        "Effect":"Allow",
        "Action":[
            "athena:CreateDataCatalog",
            "athena:DeleteDataCatalog",
            "athena:GetDataCatalog",
            "athena:GetDatabase",
            "athena:GetTableMetadata",
            "athena:ListDatabases",
            "athena:ListTableMetadata",
            "athena:UpdateDataCatalog"
```

```
        ],
        "Resource":"arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA"
      }
   ]
}
```

Example Exemple de politique pour interroger un catalogue de données spécifié

Dans la stratégie suivante, un utilisateur est autorisé à exécuter des requêtes dans le `datacatalogA` spécifié. L'utilisateur n'est pas autorisé à effectuer des tâches de gestion pour le catalogue de données lui-même, par exemple le mettre à jour ou le supprimer.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "athena:StartQueryExecution"
         ],
         "Resource":[
            "arn:aws:athena:us-east-1:123456789012:workgroup/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "athena:GetDataCatalog"
         ],
         "Resource":[
            "arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA"
         ]
      }
   ]
}
```

Example Exemple de politique pour les opérations de gestion sur un catalogue de données spécifié

Dans la stratégie suivante, un utilisateur est autorisé à créer, supprimer, obtenir des détails, et mettre à jour un catalogue de données `datacatalogA`.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateDataCatalog",
                "athena:GetDataCatalog",
                "athena:DeleteDataCatalog",
                "athena:UpdateDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA"
            ]
        }
    ]
}
```

Example Exemple de politique pour répertorier les catalogues de données

La stratégie suivante autorise tous les utilisateurs à répertorier tous les catalogues de données :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListDataCatalogs"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Exemple de politique pour les opérations de métadonnées sur les catalogues de données

La stratégie suivante autorise les opérations de métadonnées sur les catalogues de données :

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

```
            "Effect": "Allow",
            "Action": [
                "athena:GetDatabase",
                "athena:GetTableMetadata",
                "athena:ListDatabases",
                "athena:ListTableMetadata"
            ],
            "Resource": "*"
        }
    ]
}
```

Gestion des sources de données

Vous pouvez utiliser la page Data Sources (Sources de données) de la console Athena pour gérer les sources de données que vous créez.

Pour afficher une source de données

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le panneau de navigation, choisissez Data sources (Sources de données).
4. Dans la liste des sources de données, choisissez le nom de la source de données à afficher.

> **Note**
>
> Les éléments de la colonne Nom de la source de données correspondent aux résultats de l'action API ListDataCatalogs et de la commande CLI list-data-catalogs.

Pour modifier une source de données

1. Sur la page Data sources (Sources de données), suivez une des procédures suivantes :
   - Sélectionnez le bouton près du nom du catalogue, puis choisissez Actions, Edit (Modifier).
   - Choisissez le nom de la source de données. Ensuite, sur la page des détails, sélectionnez Actions, Edit (Modifier).
2. Sur la page Edit (Modifier), vous pouvez choisir une fonction Lambda différente pour la source de données, modifier la description ou ajouter des identifications personnalisées. Pour en savoir plus sur les identifications, consultez Étiquetage des ressources Athena.
3. Choisissez Enregistrer.
4. Pour modifier votre source de données AwsDataCatalog, choisissez le lien AwsDataCatalog pour ouvrir sa page de détails. Sur la page de détails, choisissez le lien vers la console AWS Glue où vous pouvez modifier votre catalogue.

Partage d'une source de données

Pour plus d'informations sur le partage des sources de données, visitez les liens suivants.

- Pour connaître les sources de données non compatibles avec Hive et basées sur Lambda, consultez Activation des requêtes fédérées entre comptes.
- Pour connaître les AWS Glue Data Catalog, consultez Accès entre comptes aux catalogues de données AWS Glue.

Pour supprimer une source de données

1. Sur la page Data sources (Sources de données), suivez une des procédures suivantes :
   - Sélectionnez le bouton à côté du nom du catalogue, puis choisissez Actions, Delete (Supprimer).

- Choisissez le nom de la source de données puis, sur la page de détails, choisissez Actions, Delete (Supprimer).

> **Note**
>
> AwsDataCatalog est la source de données par défaut de votre compte et ne peut pas être supprimé.

Soyez conscients que lorsque vous supprimez une source de données, son catalogue de données, ses tables et ses vues sont supprimés de l'éditeur de requêtes. Les requêtes enregistrées qui utilisaient la source de données supprimée ne s'exécuteront plus dans Athena.

2. Pour confirmer la suppression, saisissez le nom de la source de données, puis choisissez Delete (Supprimer).

Utilisation d'Amazon DataZone dans Athena

Vous pouvez utiliser Amazon DataZone pour partager, rechercher et découvrir des données à grande échelle au-delà des frontières de l'organisation. DataZone simplifie votre expérience grâce à des services d'analytique AWS tels qu'Athena, AWS Glue et AWS Lake Formation. Par exemple, si vous disposez de pétaoctets de données provenant de différentes sources de données, vous pouvez utiliser Amazon DataZone pour créer des groupes de personnes, de données et d'outils basés sur des cas d'utilisation métier. Pour plus d'informations, veuillez consulter Qu'est-ce que Amazon DataZone ?.

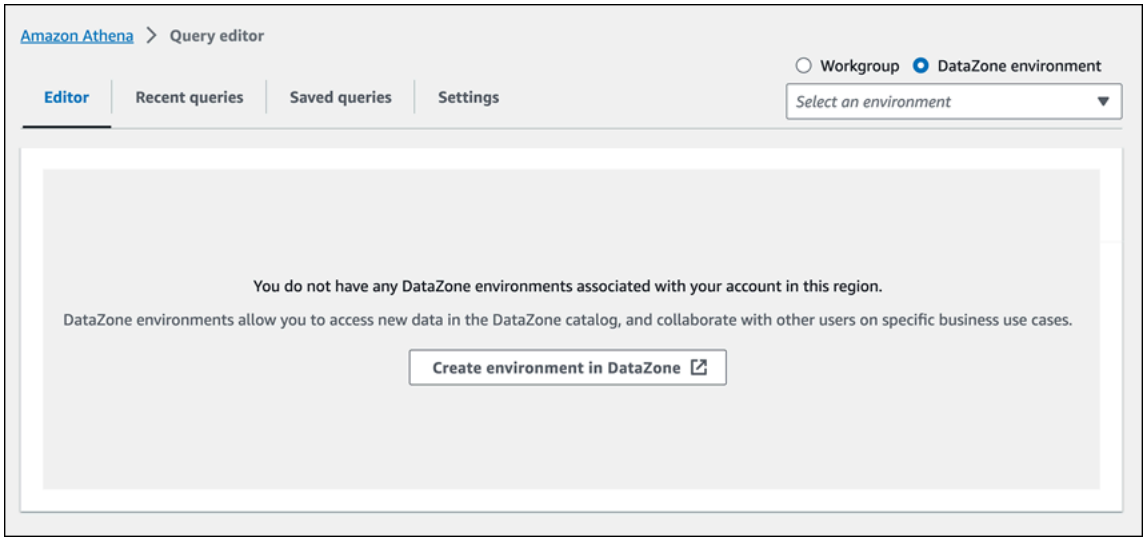
Dans Athena, vous pouvez utiliser l'éditeur de requêtes pour accéder aux environnements DataZone et les interroger. Un environnement DataZone spécifie une combinaison de projet et de domaine DataZone. Lorsque vous utilisez un environnement DataZone depuis la console Athena, vous endossez le rôle IAM de l'environnement DataZone et vous ne voyez que les bases de données et les tables appartenant à cet environnement. Les autorisations sont déterminées par les rôles que vous spécifiez dans DataZone.

Dans Athena, vous pouvez utiliser le sélecteur d'environnement DataZone sur la page de l'éditeur de requêtes pour choisir un environnement DataZone.

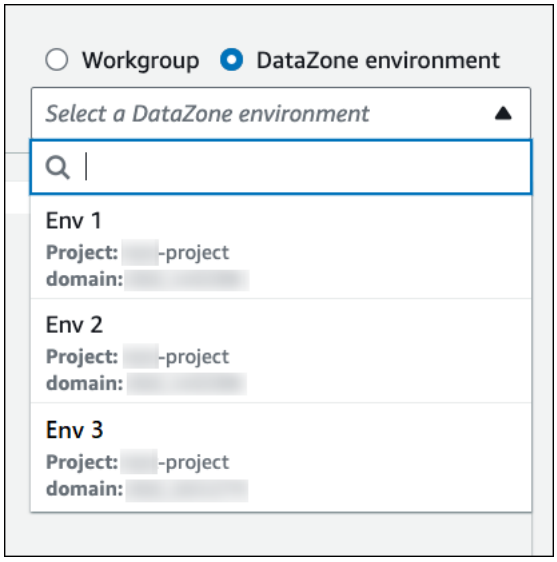
Pour ouvrir un environnement DataZone dans Athena

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans le coin supérieur droit de la console Athena, à côté de Groupe de travail, choisissez Environnement DataZone.

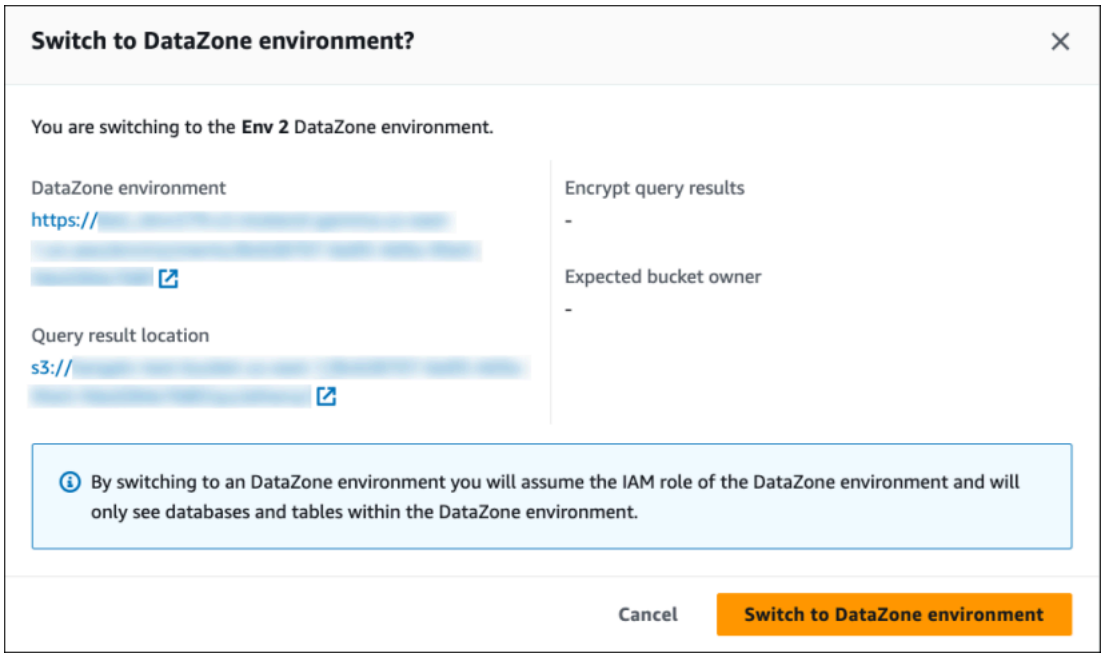
> **Note**
>
> L'option Environnement DataZone n'est présente que lorsqu'un ou plusieurs domaines sont disponibles dans DataZone.



3. Utilisez le sélecteur Environnement DataZone pour choisir un environnement DataZone.



4. Dans la boîte de dialogue Basculer vers l'environnement DataZone, vérifiez que l'environnement est celui que vous souhaitez, puis choisissez Basculer vers l'environnement DataZone.



Pour plus d'informations sur la prise en main de DataZone et Athena, veuillez consulter le didacticiel Getting started dans le Guide de l'utilisateur Amazon DataZone (langue française non garantie).

Connexion à Amazon Athena avec les pilotes ODBC et JDBC

Pour explorer et visualiser vos données avec des outils de business intelligence, téléchargez, installez et configurez un pilote ODBC (Open DataBase Connectivity) ou JDBC (Java DataBase Connectivity).

Rubriques

- Connexion à Amazon Athena avec JDBC
- Connexion à Amazon Athena avec le pilote ODBC

Consultez également les rubriques suivantes du centre de AWS connaissances et du blog AWS Big Data :

- Comment utiliser les informations d'identification de mon rôle IAM ou passer à un autre rôle IAM lors de la connexion à Athena à l'aide du pilote JDBC ?

- Configuration de la confiance entre ADFS et Active Directory AWS et utilisation des informations d'identification Active Directory pour se connecter à Amazon Athena avec le pilote ODBC

Connexion à Amazon Athena avec JDBC

Amazon Athena propose deux pilotes JDBC, versions 2.x et 3.x. Le pilote Athena JDBC 3.x est le pilote de nouvelle génération offrant de meilleures performances et une meilleure compatibilité. Le pilote JDBC 3.x prend en charge la lecture des résultats de requêtes directement depuis Amazon S3, ce qui améliore les performances des applications consommant des résultats de requêtes volumineux. Le nouveau pilote comporte également moins de dépendances tierces, ce qui facilite l'intégration aux outils de BI et aux applications personnalisées. Dans la plupart des cas, vous pouvez utiliser le nouveau pilote en apportant peu ou pas de modifications à la configuration existante.

- Pour télécharger le pilote JDBC 3.x, veuillez consulter Pilote Athena JDBC 3.x.
- Pour télécharger le pilote JDBC 2.x, veuillez consulter Pilote JDBC 2.x d'Athena.

Rubriques

Pilote Athena JDBC 3.x

Vous pouvez utiliser le pilote Athena JDBC pour vous connecter à Amazon Athena à partir de nombreux outils clients SQL tierce partie et d'applications personnalisées.

Configuration requise

- Environnement d'exécution Java 8 (ou supérieur)
- Au moins 20 Mo d'espace disque disponible

Considérations et restrictions

Voici quelques considérations et limitations relatives au pilote Athena JDBC 3.x.

- Journalisation : le pilote 3.x s'accompagne de SLF4J, qui est une couche d'abstraction permettant d'utiliser un ou plusieurs systèmes de journalisation lors de l'exécution.

- Chiffrement : lors de l'utilisation de l'outil de récupération Amazon S3 avec l'option de chiffrement `CSE_KMS`, le client Amazon S3 ne peut pas déchiffrer les résultats stockés dans un compartiment Amazon S3. Si vous avez besoin d'un chiffrement `CSE_KMS`, vous pouvez continuer à utiliser le récupérateur de streaming. La prise en charge du chiffrement `CSE_KMS` avec le récupérateur Amazon S3 est prévue.

Téléchargement du pilote JDBC 3.x

Cette section contient les informations relatives au téléchargement et à la licence du pilote JDBC 3.x.

> **Important**
>
> Lorsque vous utilisez le pilote JDBC 3.x, veillez à respecter les exigences suivantes :
>
> - Ouvrez le port 444 – Conservez ouvert le port 444, utilisé par Athena pour diffuser les résultats de requête, pour le trafic sortant. Lorsque vous utilisez un PrivateLink point de terminaison pour vous connecter à Athena, assurez-vous que le groupe de sécurité attaché au PrivateLink point de terminaison est ouvert au trafic entrant sur le port 444.
> - athena : GetQueryResultsStream policy — Ajoutez l'action de `athena:GetQueryResultsStream` stratégie aux principaux IAM qui utilisent le pilote JDBC. Cette action de politique n'est pas exposée directement avec l'API. Elle est uniquement utilisée avec les pilotes ODBC et JDBC dans le cadre de la prise en charge des résultats de streaming. Pour un exemple de politique, consultez AWS politique gérée : AWSQuicksightAthenaAccess.

Pour télécharger le pilote JDBC 3.x Amazon Athena, cliquez sur les liens suivants.

Uber jar du pilote JDBC

Le téléchargement suivant regroupe le pilote et toutes ses dépendances dans le même fichier `.jar`. Ce téléchargement est couramment utilisé pour les clients SQL tiers.

3.2.0 uber jar uber jar

Lean jar du pilote JDBC

Le téléchargement suivant est un fichier `.zip` qui contient le Lean `.jar` du pilote et des `.jar` fichiers distincts pour les dépendances du pilote. Ce téléchargement est généralement utilisé pour les applications personnalisées susceptibles de présenter des dépendances en conflit avec celles

utilisées par le pilote. Ce téléchargement est utile si vous souhaitez choisir les dépendances du pilote à inclure dans le Lean Jar et celles à exclure si votre application personnalisée en contient déjà une ou plusieurs.

3.2.0 pot maigre pot maigre

Licence

Le lien suivant contient le contrat de licence du pilote JDBC 3.x.

Licence

Rubriques

- Premiers pas avec le pilote JDBC 3.x
- Paramètres de connexion JDBC 3.x Amazon Athena
- Autre configuration JDBC 3.x
- Notes de mise à jour d'Amazon Athena JDBC 3.x
- Versions précédentes du pilote Athena JDBC 3.x

Premiers pas avec le pilote JDBC 3.x

Utilisez les informations de cette section pour commencer à utiliser le pilote JDBC 3.x Amazon Athena.

Rubriques

- Instructions d'installation
- Exécution du pilote
- Configuration du pilote
- Mise à niveau depuis le pilote Athena JDBC v2

Instructions d'installation

Vous pouvez utiliser le pilote JDBC 3.x dans une application personnalisée ou d'un client SQL tiers.

Dans une application personnalisée

Téléchargez le fichier `.zip` contenant le fichier jar du pilote et ses dépendances. Chaque dépendance possède son propre fichier `.jar`. Ajoutez le pilote jar en tant que dépendance dans

votre application personnalisée. Ajoutez de manière sélective les dépendances du fichier jar du pilote selon que vous les avez déjà ajoutées ou non à votre application à partir d'une autre source.

Dans un client SQL tiers

Téléchargez le fichier uber jar du pilote et ajoutez-le au client SQL tiers en suivant les instructions pour ce client.

Exécution du pilote

Pour exécuter le pilote, vous pouvez utiliser une application personnalisée ou un client SQL tiers.

Dans une application personnalisée

Utilisez l'interface JDBC pour interagir avec le pilote JDBC à partir d'un programme. Le code suivant montre un exemple d'application Java personnalisée.

```
public static void main(String args[]) throws SQLException {
    Properties connectionParameters = new Properties();
    connectionParameters.setProperty("Workgroup", "primary");
    connectionParameters.setProperty("Region", "us-east-2");
    connectionParameters.setProperty("Catalog", "AwsDataCatalog");
    connectionParameters.setProperty("Database","sampledatabase");
    connectionParameters.setProperty("OutputLocation","s3://DOC-EXAMPLE-BUCKET");
    connectionParameters.setProperty("CredentialsProvider","DefaultChain");
    String url = "jdbc:athena://";
    AthenaDriver driver = new AthenaDriver();
    Connection connection = driver.connect(url, connectionParameters);
    Statement statement = connection.createStatement();
    String query = "SELECT * from sample_table LIMIT 10";
    ResultSet resultSet = statement.executeQuery(query);
    printResults(resultSet); // A custom-defined method for iterating over a
                             // result set and printing its contents
}
```

Dans un client SQL tiers

Suivez la documentation du client SQL que vous utilisez. Généralement, vous utilisez l'interface utilisateur graphique du client SQL pour saisir et envoyer la requête, et les résultats de la requête sont affichés dans la même interface.

Configuration du pilote

Vous pouvez utiliser les paramètres de connexion pour configurer le pilote Amazon Athena JDBC. Pour les paramètres de connexion pris en charge, veuillez consulter Paramètres de connexion JDBC 3.x Amazon Athena.

Dans une application personnalisée

Pour définir les paramètres de connexion du pilote JDBC d'une application personnalisée, effectuez l'une des actions suivantes :

- Ajoutez les noms des paramètres et leurs valeurs à un objet `Properties`. Lorsque vous appelez `Connection#connect`, transmettez cet objet avec l'URL. Pour un exemple, veuillez consulter l'exemple d'application Java dans Exécution du pilote.
- Dans la chaîne de connexion (l'URL), utilisez le format suivant pour ajouter les noms des paramètres et leurs valeurs directement après le préfixe du protocole.

  ```
  <parameterName>=<parameterValue>;
  ```

  Utilisez un point-virgule à la fin de chaque paire nom de paramètre/valeur de paramètre, et ne laissez aucun espace après le point-virgule, comme dans l'exemple suivant.

  ```
  String url = "jdbc:athena://WorkGroup=primary;Region=us-east-1;...;";AthenaDriver
   driver = new AthenaDriver();Connection connection = driver.connect(url, null);
  ```

  > **Note**
  >
  > Si un paramètre est spécifié à la fois dans la chaîne de connexion et dans l'objet `Properties`, la valeur de la chaîne de connexion est prioritaire. Il n'est pas recommandé de spécifier le même paramètre aux deux endroits.

- Ajoutez les valeurs des paramètres en tant qu'arguments aux méthodes de `AthenaDataSource`, comme dans l'exemple suivant.

  ```
  AthenaDataSource dataSource = new AthenaDataSource();
      dataSource.setWorkGroup("primary");
      dataSource.setRegion("us-east-2");
      ...
      Connection connection = dataSource.getConnection();
  ```

```
...
```

Dans un client SQL tiers

Suivez les instructions du client SQL que vous utilisez. Généralement, le client fournit une interface utilisateur graphique pour saisir les noms des paramètres et leurs valeurs.

Mise à niveau depuis le pilote Athena JDBC v2

La plupart des paramètres de connexion JDBC version 3 sont rétrocompatibles avec le pilote JDBC version 2 (Simba). Cela signifie qu'une chaîne de connexion version 2 peut être réutilisée avec la version 3 du pilote. Certains paramètres de connexion ont toutefois changé. Ces modifications sont décrites ici. Lorsque vous effectuez une mise à niveau vers le pilote JDBC version 3, mettez à jour votre configuration existante si nécessaire.

Classe de pilote

Certains outils de BI vous demandent de fournir la classe de pilote à partir du fichier `.jar` du pilote JDBC. La plupart des outils trouvent automatiquement cette classe. Le nom entièrement qualifié de la classe dans le pilote de la version 3 est `com.amazon.athena.jdbc.AthenaDriver`. Dans le pilote version 2, la classe était `com.simba.athena.jdbc.Driver`.

Chaîne de connexion

Le pilote version 3 utilise `jdbc:athena://` le protocole au début de l'URL de la chaîne de connexion JDBC. Le pilote version 3 prend également en charge le protocole version 2`jdbc:awsathena://`, mais son utilisation est déconseillée. Pour éviter des comportements non définis, la version 3 n'accepte pas les chaînes de connexion commençant par `jdbc:awsathena://` si la version 2 (ou tout autre pilote acceptant les chaînes de connexion commençant par`jdbc:awsathena://`) a été enregistrée auprès de la DriverManagerclasse.

Fournisseurs d'informations d'identification

Le pilote version 2 utilise des noms complets pour identifier les différents fournisseurs d'informations d'identification (par exemple `com.simba.athena.amazonaws.auth.DefaultAWSCredentialsProviderChain`). Le pilote version 3 utilise des noms plus courts (par exemple `DefaultChain`). Les nouveaux noms sont décrits dans les sections correspondantes pour chaque fournisseur d'informations d'identification.

Les fournisseurs d'informations d'identification personnalisés écrits pour le pilote version 2 doivent être modifiés pour que le pilote version 3 implémente l'AwsCredentialsProviderinterface à partir de la nouvelle AWSCredentialsProviderinterface AWS SDK for Java au lieu de l'interface précédente AWS SDK for Java.

Le n'`PropertiesFileCredentialsProvider`est pas pris en charge dans le pilote JDBC 3.x. Le fournisseur a été utilisé dans le pilote JDBC 2.x mais il appartient à la version précédente du SDK pour AWS Java dont le support approche de la fin. Pour obtenir les mêmes fonctionnalités dans le pilote JDBC 3.x, utilisez plutôt le Informations d'identification du profil de configuration AWS fournisseur.

Niveau de journalisation

Le tableau suivant montre les différences entre les paramètres `LogLevel` des pilotes JDBC version 2 et version 3.

Version du pilote JDBC	Nom du paramètre	Type de paramètre	Valeur par défaut	Valeurs possibles	Exemple de chaîne de connexion
v2	`LogLevel`	Facultatif	0	0-6	`LogLevel=6;`
v3	`LogLevel`	Facultatif	TRACE	OFF, ERROR, WARN, INFO, DEBUG, TRACE	`LogLevel=INFO;`

Récupération de l'ID de requête

Dans le pilote version 2, vous décompressez une instance `Statement` dans `com.interfaces.core.IStatementQueryInfoProvider`, une interface dotée de deux méthodes : `#getPReparedQueryId` et `#getQueryId`. Vous pouvez utiliser ces méthodes pour obtenir l'ID d'exécution d'une requête déjà exécutée.

Dans le pilote version 3, vous décompressez les instances `Statement`, `PreparedStatement` et `ResultSet` de l'interface `com.amazon.athena.jdbc.AthenaResultSet`. L'interface dispose d'une seule méthode : `#getQueryExecutionId`.

Paramètres de connexion JDBC 3.x Amazon Athena

Les paramètres de connexion pris en charge sont divisés ici en trois sections : Paramètres de connexion de base, Paramètres de connexion avancés et Paramètres de connexion pour l'authentification. Les sections « Paramètres de connexion avancés » et « Paramètres de connexion d'authentification » comportent des sous-sections qui regroupent les paramètres associés.

Rubriques

- Paramètres de connexion de base
- Paramètres de connexion avancés
- Paramètres de connexion pour l'authentification

Paramètres de connexion de base

Les sections suivantes décrivent les paramètres de connexion de base du pilote JDBC 3.x.

Région

Les requêtes Région AWS Where seront exécutées. Pour obtenir la liste des régions, veuillez consulter Amazon Athena endpoints and quotas (langue française non garantie).

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Région	AwsRegion (obsolète)	Obligatoire (mais si ce n'est pas le cas, la recherche sera effectuée à l'aide du DefaultAwsRegionProviderChain)	none

Catalogue

Catalogue qui contient les bases de données et les tables auxquelles le pilote va accéder. Pour plus d'informations sur les catalogues, consultez DataCatalog.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Catalogue	none	Facultatif	AwsDataCatalog

Base de données

Base de données dans laquelle les requêtes seront exécutées. Les tables qui ne sont pas explicitement qualifiées par un nom de base de données sont résolues dans cette base de données. Pour plus d'informations sur les bases de données, veuillez consulter Database.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Database (Base de données)	Schema (Schéma)	Facultatif	default

WorkGroup

Groupe de travail dans lequel les requêtes seront exécutées. Pour plus d'informations sur les groupes de travail, consultez WorkGroup.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
WorkGroup	none	Facultatif	primary

Emplacement de sortie

Spécifie l'emplacement dans Amazon S3 où les résultats de votre requête seront stockés. Pour plus d'informations sur l'emplacement de sortie, consultez ResultConfiguration.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
OutputLocation	S3 OutputLocation (obsolète)	Obligatoire (sauf si le groupe de travail spécifie un emplacement de sortie)	none

Paramètres de connexion avancés

Les sections suivantes décrivent les paramètres de connexion avancés pour le pilote JDBC 3.x.

Rubriques

Paramètres de chiffrement du résultat

Notez les points suivants :

- La AWS KMS clé doit être spécifiée quand `EncryptionOption` est `SSE_KMS` ou`CSE_KMS`.
- La AWS KMS clé ne peut pas être spécifiée lorsqu'elle ne l'`EncryptionOption`est pas ou quand elle l'`EncryptionOption`est`SSE_S3`.

Options de chiffrement

Le type de chiffrement à utiliser pour les résultats des requêtes lorsqu'ils sont stockés dans Amazon S3. Pour plus d'informations sur le chiffrement des résultats des requêtes, consultez EncryptionConfigurationle manuel Amazon Athena API Reference.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeurs possibles
EncryptionOption	S3 OutputEnc Option (obsolète)	Facultatif	none	SSE_S3, SSE_KMS, CSE_KMS

Clé KMS

L'ARN ou l'ID de la clé KMS, si `SSE_KMS` ou `CSE_KMS` est choisi comme option de chiffrement. Pour plus d'informations, consultez EncryptionConfigurationle manuel Amazon Athena API Reference.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
KmsKey	S3 OutputEnc KMSkey (obsolète)	Facultatif	none

Paramètres de récupération des résultats

Récupérateur de résultats

Le récupérateur qui sera utilisé pour télécharger les résultats de la requête.

Le récupérateur de résultats par défaut, `S3`, télécharge les résultats des requêtes directement depuis Amazon S3 sans utiliser les API Athena. Il s'agit de l'option la plus rapide dans la plupart des cas. Cette option n'est pas disponible si les résultats de votre requête sont chiffrés avec `CSE_KMS` ou si la politique qui permet à l'utilisateur d'accéder aux résultats de la requête n'autorise que les appels d'Athéna en utilisant `s3:CalledVia`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeurs possibles
ResultFetcher	none	Facultatif	S3	S3 GetQueryResults, GetQueryResultsStream

> **Note**
>
> Dans le pilote JDBC 2.x, le `UseResultsetStreaming = 1` paramètre configure le pilote pour utiliser l'API de streaming du jeu de résultats. Dans le pilote JDBC 3.x, le paramètre équivalent est. `ResultFetcher=GetQueryResultsStream`

Taille d'extraction

La valeur de ce paramètre est utilisée comme valeur minimale pour les tampons internes et comme taille de page cible lors de la récupération des résultats. La valeur 0 (zéro) signifie que le pilote doit utiliser ses valeurs par défaut, comme décrit ci-dessous. La valeur maximale est 1 000 000.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
FetchSize	RowsToFetchPerBlock (obsolète)	Facultatif	0

- Le récupérateur `GetQueryResults` utilisera toujours une taille de page de 1 000, qui est la valeur maximale prise en charge par l'appel d'API. Lorsque la taille de récupération est supérieure à 1 000, plusieurs appels d'API successifs sont effectués pour remplir la mémoire tampon au-dessus du minimum.
- Le récupérateur `GetQueryResultsStream` utilisera la taille de lecture configurée comme taille de page, soit 10 000 par défaut.
- Le récupérateur `S3` utilisera la taille de lecture configurée comme taille de page, soit 10 000 par défaut.

Paramètres de réutilisation des résultats des requêtes

Activer la réutilisation des résultats

Indique si les résultats précédents de la même requête peuvent être réutilisés lors de l'exécution d'une requête. Pour plus d'informations sur la réutilisation des résultats de requête, consultez ResultReuseByAgeConfiguration.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
EnableResultReuseByAge	none	Facultatif	FALSE

Âge maximum de réutilisation des résultats

L'âge maximum, en minutes, d'un résultat de requête précédent qu'Athena doit envisager de réutiliser. Pour plus d'informations sur l'âge maximal de réutilisation des résultats, voir ResultReuseByAgeConfiguration.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
MaxResultReuseAgeInMinutes	none	Facultatif	60

Paramètres d'interrogation pour l'exécution des requêtes

Intervalle d'interrogation minimal pour l'exécution des requêtes

La durée minimale, en millisecondes, à attendre avant d'interroger Athena sur l'état d'exécution de la requête.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
MinQueryExecutionPollingIntervalMillis	MinQueryExecutionPollingInterval (obsolète)	Facultatif	100

Intervalle d'interrogation maximal pour l'exécution des requêtes

La durée maximale, en millisecondes, à attendre avant d'interroger Athena sur l'état d'exécution de la requête.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
MaxQueryExecutionPollingIntervalMillis	MaxQueryExecutionPollingInterval (obsolète)	Facultatif	5000

Multiplicateur d'intervalle d'interrogation pour l'exécution des requêtes

Le facteur d'augmentation de la période d'interrogation. Par défaut, l'interrogation commence par la valeur de `MinQueryExecutionPollingIntervalMillis` et double à chaque interrogation jusqu'à atteindre la valeur de `MaxQueryExecutionPollingIntervalMillis`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
QueryExecutionPollingIntervalMultiplier	none	Facultatif	2

Paramètres de remplacement du point de terminaison

Remplacement du point de terminaison Athena

Le point de terminaison que le pilote utilisera pour effectuer des appels d'API à Athena.

Notez les points suivants :

- Si les protocoles `https://` ou `http://` ne sont pas spécifiés dans l'URL fournie, le pilote insère le préfixe `https://`.
- Si ce paramètre n'est pas spécifié, le pilote utilise un point de terminaison par défaut.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AthenaEndpoint	EndpointOverride (obsolète)	Facultatif	none

Remplacement du point de terminaison du service de streaming Athena

Le point de terminaison que le pilote utilisera pour télécharger les résultats de la requête lorsqu'il utilisera le service de streaming Athena. Le service de streaming Athena est disponible sur le port 444.

Notez les points suivants :

- Si les protocoles `https://` ou `http://` ne sont pas spécifiés dans l'URL fournie, le pilote insère le préfixe `https://`.
- Si aucun port n'est spécifié dans l'URL fournie, le pilote insère le port 444 du service de streaming.
- Si le paramètre `AthenaStreamingEndpoint` n'est pas spécifié, le pilote utilise le remplacement `AthenaEndpoint`. Si ni le remplacement `AthenaStreamingEndpoint` ni le remplacement `AthenaEndpoint` ne sont spécifiés, le pilote utilise un point de terminaison de streaming par défaut.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AthenaStreamingEndpoint	StreamingEndpointOverride (obsolète)	Facultatif	none

LakeFormation remplacement du point de terminaison

Point de terminaison que le pilote utilisera pour le service Lake Formation lorsqu'il utilisera l'API AWS Lake Formation AssumeDecoratedRoleWithSAML pour récupérer des informations d'identification temporaires. Si ce paramètre n'est pas spécifié, le pilote utilise un point de terminaison Lake Formation par défaut.

Notez les points suivants :

- Si les protocoles `https://` ou `http://` ne sont pas spécifiés dans l'URL fournie, le pilote insère le préfixe `https://`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
LakeFormationEndpoint	LfEndpointOverride (obsolète)	Facultatif	none

Remplacement du point de terminaison S3

Le point de terminaison que le pilote utilisera pour télécharger les résultats de la requête lorsqu'il utilise le récupérateur Amazon S3. Si ce paramètre n'est pas spécifié, le pilote utilise un point de terminaison Amazon S3 par défaut.

Notez les points suivants :

- Si les protocoles `https://` ou `http://` ne sont pas spécifiés dans l'URL fournie, le pilote insère le préfixe `https://`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
S3Endpoint	Aucun	Facultatif	none

Remplacement du point de terminaison STS

Point de terminaison que le pilote utilisera pour le AWS STS service lorsqu'il utilisera l'API AWS STS AssumeRoleWithSAML pour récupérer des informations d'identification temporaires. Si ce paramètre n'est pas spécifié, le pilote utilise un point de AWS STS terminaison par défaut.

Notez les points suivants :

- Si les protocoles `https://` ou `http://` ne sont pas spécifiés dans l'URL fournie, le pilote insère le préfixe `https://`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
StsEndpoint	StsEndpointOverride(obsolète)	Facultatif	none

Paramètres de configuration du proxy

Hôte proxy

L'URL de l'hôte proxy. Utilisez ce paramètre si vous souhaitez que les requêtes Athena passent par un proxy.

> **Note**
>
> Assurez-vous d'inclure le protocole `https://` ou `http://` au début de l'URL de `ProxyHost`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
ProxyHost	none	Facultatif	none

Port proxy

Le port à utiliser sur l'hôte proxy. Utilisez ce paramètre si vous souhaitez que les requêtes Athena passent par un proxy.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
ProxyPort	none	Facultatif	none

Nom d'utilisateur du proxy

Le nom d'utilisateur permettant de s'authentifier sur le serveur proxy. Utilisez ce paramètre si vous souhaitez que les requêtes Athena passent par un proxy.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
ProxyUsername	ProxyUID (obsolète)	Facultatif	none

Mot de passe proxy

Le mot de passe permettant de s'authentifier sur le serveur proxy. Utilisez ce paramètre si vous souhaitez que les requêtes Athena passent par un proxy.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
ProxyPassword	ProxyPWD (obsolète)	Facultatif	none

Les hôtes exempts de proxy

Un ensemble de noms d'hôtes auxquels le pilote se connecte sans utiliser de proxy lorsque le proxy est activé (c'est-à-dire lorsque les paramètres de connexion `ProxyHost` et `ProxyPort` sont définis). Les hôtes doivent être séparés par le caractère de barre verticale (`|`) (par exemple `host1.com|host2.com`).

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
ProxyExemptHosts	NonProxyHosts	Facultatif	none

Proxy activé pour les fournisseurs d'identité

Spécifie si un proxy doit être utilisé lorsque le pilote se connecte à un fournisseur d'identité.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
ProxyEnabledForIdP	UseProxyForIdP	Facultatif	FALSE

Paramètres de journalisation

Cette section décrit les paramètres liés à la journalisation.

Niveau de journalisation

Spécifie le niveau de journalisation du pilote. Rien n'est journalisé, sauf si le paramètre `LogPath` est également défini.

> **Note**
>
> Nous vous recommandons de ne définir que le paramètre `LogPath`, sauf si vous avez des exigences particulières. La définition du seul paramètre `LogPath` permet d'activer la journalisation et d'utiliser le niveau de journalisation `TRACE` par défaut. Le niveau de journalisation `TRACE` fournit la journalisation la plus détaillée.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeurs possibles
LogLevel	none	Facultatif	TRACE	OFF, ERROR, WARN, INFO, DEBUG, TRACE

Chemin d'accès au journal

Le chemin d'accès à un répertoire de l'ordinateur qui exécute le pilote dans lequel les journaux du pilote seront stockés. Un fichier journal avec un nom unique sera créé dans le répertoire spécifié. Si cette option est définie, la journalisation du pilote est activée.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
LogPath	none	Facultatif	none

Nom de l'application

Le nom de l'application qui utilise le pilote. Si une valeur est spécifiée pour ce paramètre, elle est incluse dans la chaîne d'agent utilisateur des appels d'API que le pilote effectue à Athena.

> **Note**
>
> Vous pouvez également définir le nom de l'application en appelant `setApplicationName` sur l'objet `DataSource`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
ApplicationName	none	Facultatif	none

Test de connexion

S'il est défini sur TRUE, le pilote effectue un test de connexion chaque fois qu'une connexion JDBC est créée, même si aucune requête n'est exécutée sur la connexion.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
ConnectionTest	none	Facultatif	TRUE

> **Note**
>
> Un test de connexion envoie une requête `SELECT 1` à Athena pour vérifier que la connexion a été correctement configurée. Cela signifie que deux fichiers seront stockés dans Amazon S3 (le jeu de résultats et les métadonnées), et des frais supplémentaires peuvent s'appliquer conformément à la tarification d'Amazon Athena.

Nombre de nouvelles tentatives

Le nombre maximal de fois que le pilote doit renvoyer une requête réitérable à Athena.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
NumRetries	MaxErrorRetry (obsolète)	Facultatif	none

Paramètres de connexion pour l'authentification

Le pilote Athena JDBC 3.x prend en charge plusieurs méthodes d'authentification. Les paramètres de connexion requis dépendent de la méthode d'authentification que vous utilisez.

Rubriques

- Informations d'identification IAM
- Informations d'identification par défaut
- Informations d'identification du profil de configuration AWS
- Informations d'identification du profil d'instance
- Informations d'identification personnalisées
- Informations d'identification JWT
- Informations d'identification Azure AD
- Informations d'identification Okta
- Informations d'identification Ping
- Informations d'identification AD FS
- Informations d'identification du navigateur Azure AD
- Informations d'identification du navigateur SAML

Informations d'identification IAM

Vous pouvez utiliser vos informations d'identification IAM avec le pilote JDBC pour vous connecter à Amazon Athena en définissant les paramètres de connexion suivants.

Utilisateur

L'identifiant de votre clé d' AWS accès. Pour plus d'informations sur les clés d'accès, veuillez consulter Informations d'identification de sécuritéAWS dans le Guide de l'utilisateur IAM.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Utilisateur	AccessKeyId	Obligatoire	none

Mot de passe

L'identifiant de votre clé AWS secrète. Pour plus d'informations sur les clés d'accès, veuillez consulter Informations d'identification de sécuritéAWS dans le Guide de l'utilisateur IAM.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Mot de passe	SecretAccessKey	Facultatif	none

Jeton de session

Si vous utilisez des AWS informations d'identification temporaires, vous devez spécifier un jeton de session. Pour plus d'informations sur les informations d'identification temporaires, consultez Informations d'identification de sécurité temporaires dans le Guide de l'utilisateur IAM.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
SessionToken	none	Facultatif	none

Informations d'identification par défaut

Vous pouvez utiliser les informations d'identification par défaut que vous configurez sur votre système client pour vous connecter à Amazon Athena en définissant les paramètres de connexion suivants. Pour plus d'informations sur l'utilisation des informations d'identification par défaut, consultez Utilisation de la chaîne de fournisseurs d'informations d'identification par défaut dans le Guide du développeur AWS SDK for Java .

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur `DefaultChain`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass (obsolète)	Obligatoire	none	`DefaultChain`

Informations d'identification du profil de configuration AWS

Vous pouvez utiliser les informations d'identification stockées dans un profil de AWS configuration en définissant les paramètres de connexion suivants. AWS les profils de configuration sont généralement stockés dans des fichiers du `~/.aws` répertoire). Pour plus d'informations sur les profils de configuration AWS , veuillez consulter Use profiles dans le Guide du développeur AWS SDK for Java .

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur `ProfileCredentials`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass(obsolète)	Obligatoire	none	`ProfileCredentials`

Profile name (Nom du profil)

Le nom du profil de AWS configuration dont les informations d'identification doivent être utilisées pour authentifier la demande adressée à Athena.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
ProfileName	none	Obligatoire	none

> **Note**
>
> Le nom du profil peut également être spécifié comme valeur du paramètre `CredentialsProviderArguments`, bien que cette pratique soit obsolète.

Informations d'identification du profil d'instance

Ce type d'authentification est utilisé sur les instances Amazon EC2. Un profil d'instance est un profil attaché à une instance Amazon EC2. L'utilisation d'un fournisseur d'informations d'identification de profil d'instance délègue la gestion des AWS informations d'identification au service de métadonnées d'instance Amazon EC2. Les développeurs n'ont donc plus à stocker les informations d'identification de manière permanente sur l'instance Amazon EC2 ou à se soucier de la rotation ou de la gestion des informations d'identification temporaires.

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur `InstanceProfile`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass (obsolète)	Obligatoire	none	`InstanceProfile`

Informations d'identification personnalisées

Vous pouvez utiliser ce type d'authentification pour fournir vos propres informations d'identification à l'aide d'une classe Java qui implémente l'AwsCredentialsProviderinterface.

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur le nom de classe complet de la classe personnalisée qui implémente l'AwsCredentialsProviderinterface. Lors de l'exécution, cette classe doit se trouver sur le chemin de classe Java de l'application qui utilise le pilote JDBC.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass (obsolète)	Obligatoire	none	Le nom de classe complet de l'implémentation personnalisée de `AwsCredentialsProvider`

Arguments du fournisseur d'informations d'identification

Une liste d'arguments de chaîne de caractères séparés par des virgules pour le constructeur du fournisseur d'informations d'identification personnalisé.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
CredentialsProvideArguments	AwsCredentialsProviderArguments (obsolète)	Facultatif	none

Informations d'identification JWT

Avec ce type d'authentification, vous pouvez utiliser un jeton Web JSON (JWT) obtenu auprès d'un fournisseur d'identité externe comme paramètre de connexion pour vous authentifier auprès d'Athena. Le fournisseur d'informations d'identification externe doit déjà être fédéré avec AWS.

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur `JWT`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass (obsolète)	Obligatoire	none	`JWT`

Jeton d'identité web JWT

Le jeton JWT obtenu auprès d'un fournisseur d'identité fédéré externe. Ce jeton sera utilisé pour s'authentifier auprès d'Athéna.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
JwtWebIdentityToken	web_identity_token (obsolète)	Obligatoire	none

ARN du rôle JWT

Amazon Resource Name (ARN) du rôle à assumer. Pour plus d'informations sur l'attribution de rôles, consultez AssumeRolela référence des AWS Security Token Service API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
JwtRoleArn	role_arn (obsolète)	Obligatoire	none

Nom de la session de rôle JWT

Le nom de la session lorsque vous utilisez les informations d'identification JWT pour l'authentification. Le nom peut être n'importe quel nom que vous choisissez.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
JwtRoleSessionName	role_session_name (obsolète)	Obligatoire	none

Informations d'identification Azure AD

Un mécanisme d'authentification basé sur SAML qui permet l'authentification auprès d'Athena via le fournisseur d'identité Azure AD. Cette méthode suppose qu'une fédération a déjà été configurée entre Athena et Azure AD.

> **Note**
>
> Certains des noms de paramètres de cette section comportent des alias. Les alias sont des équivalents fonctionnels des noms de paramètres et ont été fournis à des fins de rétrocompatibilité avec le pilote JDBC 2.x. Les noms des paramètres ayant été améliorés pour suivre une convention de dénomination plus claire et plus cohérente, nous vous recommandons de les utiliser à la place des alias, qui sont devenus obsolètes.

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur `AzureAD`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass (obsolète)	Obligatoire	none	`AzureAD`

Utilisateur

L'adresse e-mail de l'utilisateur Azure AD à utiliser pour l'authentification avec Azure AD.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Utilisateur	UID (obsolète)	Obligatoire	none

Mot de passe

Le mot de passe pour l'utilisateur Azure AD.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Mot de passe	PWD (obsolète)	Obligatoire	none

ID de locataire Azure AD

L'ID de locataire de votre application Azure AD.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AzureAdTenantId	tenant_id (obsolète)	Obligatoire	none

ID client Azure AD

L'ID client de votre application Azure AD.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AzureAdClientId	client_id (obsolète)	Obligatoire	none

Secret client Azure AD

Le secret client de votre application Azure AD.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AzureAdClientSecret	client_secret (obsolète)	Obligatoire	none

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Pour plus d'informations sur les rôles ARN, consultez AssumeRolela référence de l'AWS Security Token Service API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
PreferredRole	preferred_role (obsolète)	Facultatif	none

Durée de la session de rôle

La durée de la session de rôle en secondes. Pour plus d'informations, consultez AssumeRolela référence de AWS Security Token Service l'API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
RoleSessionDuration	Duration (obsolète)	Facultatif	3600

Lake Formation activé

Spécifie s'il faut utiliser l'action d'API AssumeDecoratedRoleWithSAML Lake Formation pour récupérer des informations d'identification IAM temporaires au lieu de l'action d'API AssumeRoleWithSAML. AWS STS

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
LakeFormationEnabled	none	Facultatif	FALSE

Informations d'identification Okta

Un mécanisme d'authentification basé sur SAML qui permet l'authentification auprès d'Athena via le fournisseur d'identité Okta. Cette méthode suppose qu'une fédération a déjà été configurée entre Athena et Okta.

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur `Okta`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass (obsolète)	Obligatoire	none	`Okta`

Utilisateur

L'adresse e-mail de l'utilisateur Okta à utiliser pour l'authentification avec Okta.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Utilisateur	UID (obsolète)	Obligatoire	none

Mot de passe

Le mot de passe de l'utilisateur Okta.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Mot de passe	PWD (obsolète)	Obligatoire	none

Nom d'hôte Okta

URL de votre organisation Okta. Vous pouvez extraire le paramètre `idp_host` de l'URL de lien intégré dans votre application Okta. Pour les étapes, consultez Récupération des informations de configuration ODBC d'Okta. Le premier segment suivant `https://`, jusqu'à `okta.com` inclus, est votre hôte IdP (par exemple `trial-1234567.okta.com` pour une URL commençant par `https://trial-1234567.okta.com`).

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
OktaHostName	IdP_Host (obsolète)	Obligatoire	none

ID de l'application Okta

Saisissez l'identifiant en deux parties de votre application. Vous pouvez extraire le paramètre ID de l'URL de lien intégré dans votre application Okta. Pour les étapes, consultez Récupération des informations de configuration ODBC d'Okta. L'ID de l'application correspond aux deux derniers segments de l'URL, y compris la barre oblique au milieu. Les segments sont deux chaînes de 20 caractères contenant un mélange de chiffres et de lettres majuscules et minuscules (par exemple, `Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4`).

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
OktaAppId	App_ID (obsolète)	Obligatoire	none

Nom de l'application Okta

Le nom de votre application Okta.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
OktaAppName	App_Name (obsolète)	Obligatoire	none

Type MFA Okta

Si vous avez configuré Okta pour exiger l'authentification multifactorielle (MFA), vous devez spécifier le type d'Okta MFA et des paramètres supplémentaires en fonction du second facteur que vous souhaitez utiliser.

Le type Okta MFA est le deuxième type de facteur d'authentification (après le mot de passe) à utiliser pour s'authentifier auprès d'Okta. Les seconds facteurs pris en charge incluent les notifications push envoyées via l'application Okta Verify et les mots de passe temporaires à usage unique (TOTP) générés par Okta Verify, Google Authenticator ou envoyés par SMS. Les politiques de sécurité de chaque organisation déterminent si l'authentification multifactorielle est requise pour la connexion des utilisateurs.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeurs possibles
OktaMfaType	okta_mfa_type (obsolète)	Obligatoire, si Okta est configuré pour exiger l'authentification MFA	none	`oktaverifywithpush` , `oktaverifywithtotp` , `googleauthenticator` , `smsauthentication`

Numéro de téléphone Okta

Le numéro de téléphone auquel Okta enverra un mot de passe temporaire à usage unique par SMS lorsque le type de MFA `smsauthentication` est sélectionné. Le numéro de téléphone doit être un numéro de téléphone américain ou canadien.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
OktaPhoneNumber	okta_phone_number (obsolète)	Obligatoire, si OktaMfaType c'est le cas `smsauthentication`	none

Temps d'attente de l'authentification MFA Okta

La durée, en secondes, au cours de laquelle l'utilisateur doit accuser réception d'une notification push d'Okta avant que le pilote ne lance une exception de délai d'expiration.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
OktaMfaWaitTime	okta_mfa_wait_time (obsolète)	Facultatif	60

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Pour plus d'informations sur les rôles ARN, consultez AssumeRolela référence de l'AWS Security Token Service API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
PreferredRole	preferred_role (obsolète)	Facultatif	none

Durée de la session de rôle

La durée de la session de rôle en secondes. Pour plus d'informations, consultez AssumeRolela référence de AWS Security Token Service l'API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
RoleSessionDuration	Duration (obsolète)	Facultatif	3600

Lake Formation activé

Spécifie s'il faut utiliser l'action d'API AssumeDecoratedRoleWithSAML Lake Formation pour récupérer des informations d'identification IAM temporaires au lieu de l'action d'API AssumeRoleWithSAML. AWS STS

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
LakeFormationEnabled	none	Facultatif	FALSE

Informations d'identification Ping

Un mécanisme d'authentification basé sur SAML qui permet l'authentification auprès d'Athena à l'aide du fournisseur d'identité Ping Federate. Cette méthode suppose qu'une fédération a déjà été configurée entre Athena et Ping Federate.

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur `Ping`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass (obsolète)	Obligatoire	none	`Ping`

Utilisateur

L'adresse e-mail de l'utilisateur de Ping Federate à utiliser pour l'authentification auprès de Ping Federate.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Utilisateur	UID (obsolète)	Obligatoire	none

Mot de passe

Le mot de passe de l'utilisateur Ping Federate.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Mot de passe	PWD (obsolète)	Obligatoire	none

PingHostName

L'adresse de votre serveur Ping. Pour trouver votre adresse, rendez-vous sur l'URL suivante et consultez le champ Point de terminaison de l'application SSO.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
PingHostName	IdP_Host (obsolète)	Obligatoire	none

PingPortNumber

Le numéro de port à utiliser pour vous connecter à votre hôte IdP.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
PingPortNumber	IdP_Port (obsolète)	Obligatoire	none

PingPartnerSpId

Adresse du fournisseur de services. Pour trouver l'adresse du fournisseur de services, rendez-vous sur l'URL suivante et consultez le champ Point de terminaison de l'application SSO.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
PingPartnerSpId	Partner_SPID (obsolète)	Obligatoire	none

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Pour plus d'informations sur les rôles ARN, consultez AssumeRolela référence de l'AWS Security Token Service API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
PreferredRole	preferred_role (obsolète)	Facultatif	none

Durée de la session de rôle

La durée de la session de rôle en secondes. Pour plus d'informations, consultez AssumeRolela référence de AWS Security Token Service l'API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
RoleSessionDuration	Duration (obsolète)	Facultatif	3600

Lake Formation activé

Spécifie s'il faut utiliser l'action d'API AssumeDecoratedRoleWithSAML Lake Formation pour récupérer des informations d'identification IAM temporaires au lieu de l'action d'API AssumeRoleWithSAML. AWS STS

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
LakeFormationEnabled	none	Facultatif	FALSE

Informations d'identification AD FS

Mécanisme d'authentification basé sur SAML qui permet l'authentification auprès d'Athena à l'aide de Microsoft Active Directory Federation Services (AD FS). Cette méthode suppose que l'utilisateur a déjà configuré une fédération entre Athena et AD FS.

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur `ADFS`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass (obsolète)	Obligatoire	none	`ADFS`

Utilisateur

Adresse e-mail de l'utilisateur AD FS à utiliser pour l'authentification auprès d'AD FS.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Utilisateur	UID (obsolète)	Nécessaire pour l'authentification basée sur un formulaire. Facultatif pour l'authentification intégrée Windows.	none

Mot de passe

Le mot de passe de l'utilisateur AD FS.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
Mot de passe	PWD (obsolète)	Nécessaire pour l'authentification basée sur un formulaire. Facultatif pour l'authentification intégrée Windows.	none

Nom d'hôte ADFS

Adresse de votre serveur AD FS.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AdfsHostName	IdP_Host (obsolète)	Obligatoire	none

Numéro de port ADFS

Numéro de port à utiliser pour vous connecter à votre serveur AD FS.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AdfsPortNumber	IdP_Port (obsolète)	Obligatoire	none

Partie utilisatrice ADFS

Partie utilisatrice approuvée. Utilisez ce paramètre pour remplacer l'URL du point de terminaison de la partie utilisatrice AD FS.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AdfsRelyingParty	LoginToRP (obsolète)	Facultatif	`urn:amazon:webservices`

ADFS WIA activé

Booléen. Utilisez ce paramètre pour activer l'authentification intégrée Windows (WIA) avec AD FS.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AdfsWiaEnabled	none	Facultatif	FALSE

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Pour plus d'informations sur les rôles ARN, consultez `AssumeRole`la référence de l'AWS Security Token Service API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
PreferredRole	preferred_role (obsolète)	Facultatif	none

Durée de la session de rôle

La durée de la session de rôle en secondes. Pour plus d'informations, consultez `AssumeRole` dans la Référence d'API AWS Security Token Service .

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
RoleSessionDuration	Duration (obsolète)	Facultatif	3600

Lake Formation activé

Spécifie s'il faut utiliser l'action d'API `AssumeDecoratedRoleWithSAML`Lake Formation pour récupérer des informations d'identification IAM temporaires au lieu de l'action d'`AssumeRoleWithSAML` AWS STS API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
LakeFormationEnabled	none	Facultatif	FALSE

Informations d'identification du navigateur Azure AD

Le navigateur Azure AD est un mécanisme d'authentification basé sur SAML qui fonctionne avec le fournisseur d'identité Azure AD et prend en charge l'authentification multifactorielle. Contrairement au mécanisme d'authentification Azure AD standard, celui-ci ne nécessite pas de nom d'utilisateur, de mot de passe ou de secret client dans les paramètres de connexion. À l'instar du mécanisme d'authentification Azure AD standard, le navigateur Azure AD suppose également que l'utilisateur a déjà configuré la fédération entre Athena et Azure AD.

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur `BrowserAzureAD`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass (obsolète)	Obligatoire	none	`BrowserAzureAD`

ID de locataire Azure AD

L'ID de locataire de votre application Azure AD

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AzureAdTenantId	tenant_id (obsolète)	Obligatoire	none

ID client Azure AD

L'ID client de votre application Azure AD

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
AzureAdClientId	client_id (obsolète)	Obligatoire	none

Délai d'expiration de la réponse du fournisseur d'identité

La durée, en secondes, avant que le pilote n'arrête d'attendre la réponse SAML d'Azure AD.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
IdpResponseTimeout	idp_response_timeout (obsolète)	Facultatif	120

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Pour plus d'informations sur les rôles ARN, consultez AssumeRolela référence de l'AWS Security Token Service API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
PreferredRole	preferred_role (obsolète)	Facultatif	none

Durée de la session de rôle

La durée de la session de rôle en secondes. Pour plus d'informations, consultez AssumeRolela référence de AWS Security Token Service l'API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
RoleSessionDuration	Duration (obsolète)	Facultatif	3600

Lake Formation activé

Spécifie s'il faut utiliser l'action d'API AssumeDecoratedRoleWithSAML Lake Formation pour récupérer des informations d'identification IAM temporaires au lieu de l'action d'API AssumeRoleWithSAML. AWS STS

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
LakeFormationEnabled	none	Facultatif	FALSE

Informations d'identification du navigateur SAML

Le navigateur SAML est un plug-in d'authentification générique qui peut fonctionner avec les fournisseurs d'identité basés sur SAML et prend en charge l'authentification multifactorielle.

Fournisseur d'informations d'identification

Le fournisseur d'informations d'identification qui sera utilisé pour authentifier les requêtes adressées à AWS. Définissez la valeur de ce paramètre sur `BrowserSaml`.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut	Valeur à utiliser
CredentialsProvider	AWSCredentialsProviderClass (obsolète)	Obligatoire	none	`BrowserSaml`

URL de connexion à authentification unique

L'URL de connexion à authentification unique de votre application via le fournisseur d'identité basé sur SAML.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
SsoLoginUrl	login_url (obsolète)	Obligatoire	none

Port d'écoute

Numéro de port utilisé pour écouter la réponse SAML. Cette valeur doit correspondre à l'URL avec laquelle vous avez configuré le fournisseur d'identité basé sur SAML (par exemple `http://localhost:7890/athena`).

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
ListenPort	listen_port (obsolète)	Facultatif	7890

Délai d'expiration de la réponse du fournisseur d'identité

La durée, en secondes, avant que le pilote n'arrête d'attendre la réponse SAML d'Azure AD.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
IdpResponseTimeout	idp_response_timeout (obsolète)	Facultatif	120

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Pour plus d'informations sur les rôles ARN, consultez AssumeRolela référence de l'AWS Security Token Service API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
PreferredRole	preferred_role (obsolète)	Facultatif	none

Durée de la session de rôle

La durée de la session de rôle en secondes. Pour plus d'informations, consultez AssumeRolela référence de AWS Security Token Service l'API.

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
RoleSessionDuration	Duration (obsolète)	Facultatif	3600

Lake Formation activé

Spécifie s'il faut utiliser l'action d'API AssumeDecoratedRoleWithSAML Lake Formation pour récupérer des informations d'identification IAM temporaires au lieu de l'action d'API AssumeRoleWithSAML. AWS STS

Nom du paramètre	Alias	Type de paramètre	Valeur par défaut
LakeFormationEnabled	none	Facultatif	FALSE

Autre configuration JDBC 3.x

Les sections suivantes décrivent certains paramètres de configuration supplémentaires pour le pilote JDBC 3.x.

Délai d'expiration du réseau

Le temps d'attente, en millisecondes, pendant lequel le pilote attendra une réponse lorsqu'il fait un appel d'API à Athena. Passé ce délai, le pilote lance une exception de délai d'expiration.

Le délai d'expiration du réseau ne peut pas être défini en tant que paramètre de connexion. Pour le définir, appelez la méthode `setNetworkTimeout` sur un objet JDBC `Connection`. Cette valeur peut être modifiée pendant le cycle de vie de la connexion JDBC. La valeur par défaut de ce paramètre est `infinity`.

L'exemple suivant définit le délai d'expiration du réseau à 5 000 millisecondes.

```
...
AthenaDriver driver = new AthenaDriver();
Connection connection = driver.connect(url, connectionParameters);
connection.setNetworkTimeout(null, 5000);
...
```

Délai d'expiration de la requête

Le temps, en secondes, pendant lequel le pilote attendra la fin d'une requête sur Athena après qu'une requête ait été soumise. Passé ce délai, le pilote tente d'annuler la requête soumise et lance une exception de délai d'expiration.

Le délai d'expiration de la requête ne peut pas être défini en tant que paramètre de connexion. Pour le définir, appelez la méthode `setQueryTimeout` sur un objet JDBC `Statement`. Cette valeur peut être modifiée au cours du cycle de vie d'une instruction JDBC. La valeur par défaut de ce paramètre est `0` (zéro). La valeur de `0` signifie que les requêtes peuvent être exécutées jusqu'à ce qu'elles soient terminées (sous réserve de Service Quotas).

L'exemple suivant définit le délai d'expiration de la requête à 5 secondes.

```
...
AthenaDriver driver = new AthenaDriver();
Connection connection = driver.connect(url, connectionParameters);
Statement statement = connection.createStatement();
statement.setQueryTimeout(5);
...
```

Notes de mise à jour d'Amazon Athena JDBC 3.x

Ces notes de mise à jour fournissent des informations détaillées sur les améliorations et les correctifs apportés au pilote Amazon Athena JDBC 3.x.

3.2.0

Publié le 26/04/2022

Améliorations

- Performances des opérations de catalogue : les performances ont été améliorées pour les opérations de catalogue qui n'utilisent pas de caractères génériques.
- Modification de l'intervalle d'interrogation minimal : l'intervalle d'interrogation minimal par défaut a été modifié afin de réduire le nombre d'appels d'API que le pilote fait à Athena. Les requêtes terminées sont toujours détectées dès que possible.
- Découvrabilité des outils de BI — Le moteur a été rendu plus facilement détectable pour les outils de business intelligence.
- Mappage des types de données — Le mappage des types de données avec les types de données Athena `binary` et `struct` DDL a été amélioré. `array`
- AWS Version du SDK — La version du AWS SDK utilisée dans le pilote a été mise à jour vers la version 2.25.34.

Correctifs

- Listes de tables de catalogue fédérées : correction d'un problème en raison duquel les catalogues fédérés renvoyaient une liste de tables vide.
- GetSchemas — Correction d'un problème en raison duquel la méthode JDBC DatabaseMetaData#getSchemas récupérait les bases de données uniquement à partir du catalogue par défaut et non à partir de tous les catalogues.
- GetColumns — Correction d'un problème qui provoquait le renvoi d'un catalogue nul lorsque la méthode JDBC DatabaseMetaData#getColumns était appelée avec un nom de catalogue nul.

3.1.0

Publié le 15/02/2022

Améliorations

- Support ajouté pour l'authentification intégrée Windows Microsoft Active Directory Federation Services (AD FS) et l'authentification par formulaire.
- Pour des raisons de rétrocompatibilité avec la version 2.x, le sous-protocole `awsathena` JDBC est désormais accepté mais produit un avertissement d'obsolescence. Utilisez plutôt le `athena` sous-protocole JDBC.
- `AwsDataCatalog`est désormais la valeur par défaut pour le paramètre du catalogue, et `default` c'est la valeur par défaut pour le paramètre de base de données. Ces modifications garantissent que les valeurs correctes du catalogue et de la base de données actuels sont renvoyées au lieu de valeurs nulles.
- Conformément à la spécification JDBC, `IS_AUTOINCREMENT` et renvoie `IS_GENERATEDCOLUMN` désormais une chaîne vide au lieu de. N0
- Le type de `int` données Athena correspond désormais au même type JDBC qu'Athena plutôt qu'à. `integer` `other`
- Lorsque les métadonnées de colonne d'Athena ne contiennent pas les `scale` champs facultatifs `precision` et, le pilote renvoie désormais zéro pour les valeurs correspondantes dans une `ResultSet` colonne.
- La version du AWS SDK a été mise à jour vers la version 2.21.39.

Correctifs

- Correction d'un problème `GetQueryResultsStream` qui provoquait une exception lorsque le nombre de colonnes des résultats en texte brut d'Athéna ne correspondait pas au nombre de colonnes indiqué dans les métadonnées des résultats d'Athéna.

3.0.0

Publié le 16/11/2023

Le pilote Athena JDBC 3.x est le pilote de nouvelle génération offrant de meilleures performances et une meilleure compatibilité. Le pilote JDBC 3.x prend en charge la lecture des résultats de requêtes directement depuis Amazon S3, ce qui améliore les performances des applications consommant des résultats de requêtes volumineux. Le nouveau pilote comporte également moins de dépendances tierces, ce qui facilite l'intégration aux outils de BI et aux applications personnalisées.

Versions précédentes du pilote Athena JDBC 3.x

Nous vous recommandons vivement d'utiliser la dernière version du pilote JDBC 3.x. La dernière version du pilote contient les améliorations et correctifs les plus récents. Utilisez une ancienne version uniquement si votre application présente des incompatibilités avec la dernière version.

Uber jar du pilote JDBC

Le téléchargement suivant regroupe le pilote et toutes ses dépendances dans le même fichier `.jar`. Ce téléchargement est couramment utilisé pour les clients SQL tiers.

- 3.1.0 uber jar uber jar
- 3.0.0 uber jar uber jar

Lean jar du pilote JDBC

Le téléchargement suivant est un fichier `.zip` qui contient le Lean `.jar` du pilote et des `.jar` fichiers distincts pour les dépendances du pilote. Ce téléchargement est généralement utilisé pour les applications personnalisées susceptibles de présenter des dépendances en conflit avec celles utilisées par le pilote. Ce téléchargement est utile si vous souhaitez choisir les dépendances du pilote à inclure dans le Lean Jar et celles à exclure si votre application personnalisée en contient déjà une ou plusieurs.

- 3.1.0 bocal maigre bocal maigre

- 3.0.0 pot maigre pot maigre

Pilote JDBC 2.x d'Athena

Vous pouvez utiliser une connexion JDBC pour connecter Athena à des outils de business intelligence et à d'autres applications, telles que SQL workbench. Pour ce faire, utilisez les liens Amazon S3 de cette page pour télécharger, installer et configurer le pilote Athena JDBC 2.x. Pour obtenir des informations sur la création de l'URL de connexion JDBC, consultez le document téléchargeable Guide d'installation et de configuration du pilote JDBC. Pour plus d'informations sur les autorisations, voir Accès via les connexions JDBC et ODBC. Pour envoyer des commentaires concernant le pilote JDBC, envoyez un e-mail à athena-feedback@amazon.com. À partir de la version 2.0.24, deux versions du pilote sont disponibles : une qui inclut le AWS SDK et une qui ne l'inclut pas.

Important

Lorsque vous utilisez le pilote JDBC, veillez à respecter les exigences suivantes :

- Ouvrez le port 444 – Conservez ouvert le port 444, utilisé par Athena pour diffuser les résultats de requête, pour le trafic sortant. Lorsque vous utilisez un PrivateLink point de terminaison pour vous connecter à Athena, assurez-vous que le groupe de sécurité attaché au PrivateLink point de terminaison est ouvert au trafic entrant sur le port 444. Si le port 444 est bloqué, il est possible que vous receviez le message d'erreur [Simba][AthenaJDBC](100123) Une erreur s'est produite. Exception during column initialization (Exception pendant l'initialisation de la colonne).
- athena : GetQueryResultsStream policy — Ajoutez l'action de `athena:GetQueryResultsStream` stratégie aux principaux IAM qui utilisent le pilote JDBC. Cette action de politique n'est pas exposée directement avec l'API. Elle est uniquement utilisée avec les pilotes ODBC et JDBC dans le cadre de la prise en charge des résultats de streaming. Pour un exemple de politique, consultez AWS politique gérée : AWSQuicksightAthenaAccess.
- Utilisation du pilote JDBC pour plusieurs catalogues de données – Pour utiliser le pilote JDBC pour plusieurs catalogues de données avec Athena (par exemple, lors de l'utilisation d'un métastore Hive externe ou de requêtes fédérées), incluez `MetadataRetrievalMethod=ProxyAPI` dans votre chaîne de connexion JDBC.

- Pilotes 4.1 – À partir de 2023, la prise en charge des pilotes pour JDBC version 4.1 sera interrompue. Aucune autre mise à jour ne sera publiée. Si vous utilisez un pilote JDBC 4.1, la migration vers le pilote 4.2 est fortement recommandée.

Pilote JDBC 2.x avec SDK AWS

La version 2.1.5 du pilote JDBC est conforme à la norme de données JDBC API 4.2 et nécessite le JDK 8.0 ou une version ultérieure. Pour plus d'informations sur la vérification de la version de l'Environnement d'exécution Java (JRE) que vous utilisez, consultez la documentation Java.

Utilisez le lien suivant pour télécharger le fichier `.jar` du pilote JDBC 4.2.

- https://downloads.athena.us-east-1.amazonaws.com/drivers/JDBC/SimbaAthenaJDBC-2.1.5.1000/AthenaJDBC42-2.1.5.1000.jar

Le `.zip` fichier à télécharger ci-dessous contient le `.jar` fichier pour JDBC 4.2 et inclut le AWS SDK ainsi que la documentation, les notes de publication, les licences et les accords qui l'accompagnent.

- SimbaAthena

Pilote JDBC 2.x sans kit SDK AWS

La version 2.1.5 du pilote JDBC est conforme à la norme de données JDBC API 4.2 et nécessite le JDK 8.0 ou une version ultérieure. Pour plus d'informations sur la vérification de la version de l'Environnement d'exécution Java (JRE) que vous utilisez, consultez la documentation Java.

Cliquez sur le lien suivant pour télécharger le `.jar` fichier du pilote JDBC 4.2 sans le AWS SDK.

- https://downloads.athena.us-east-1.amazonaws.com/drivers/JDBC/SimbaAthenaJDBC-2.1.5.1001/AthenaJDBC42-2.1.5.1001.jar

Le téléchargement du fichier `.zip` suivant contient le fichier `.jar` pour JDBC 4.2 et la documentation, les notes de mise à jour, les licences et les accords qui l'accompagnent. Il n'inclut pas le AWS SDK.

- SimbaAthena

Notes de mise à jour du pilote JDBC 2.x, contrat de licence et mentions légales

Après avoir téléchargé la version dont vous avez besoin, consultez les notes de mise à jour ainsi que le Contrat de licence et les Mentions légales.

- Notes de mise à jour
- Contrat de licence
- Avis
- Licences tierces

Documentation du pilote JDBC 2.x

Téléchargez la documentation suivante pour le pilote :

- Guide de configuration et d'installation du pilote JDBC. Utilisez ce guide pour installer et configurer le pilote.
- Guide de migration des pilotes JDBC. Utilisez ce guide pour migrer des versions antérieures vers la version actuelle.

Connexion à Amazon Athena avec le pilote ODBC

Amazon Athena propose deux pilotes ODBC, versions 1.x et 2.x. Le pilote Athena ODBC 2.x est une nouvelle alternative compatible avec les systèmes Linux, macOS ARM, macOS Intel et Windows 64 bits. Le pilote Athena 2.x prend en charge tous les plug-ins d'authentification pris en charge par le pilote ODBC 1.x, et presque tous les paramètres de connexion sont rétrocompatibles.

- Pour télécharger le pilote ODBC 2.x, consultez Amazon Athena ODBC 2.x.
- Pour télécharger le pilote ODBC 1.x, consultez Pilote ODBC 1.x d'Athena.

Rubriques

- Amazon Athena ODBC 2.x
- Pilote ODBC 1.x d'Athena
- Utilisation du connecteur Amazon Athena Power BI

Amazon Athena ODBC 2.x

Vous pouvez utiliser une connexion ODBC pour vous connecter à Amazon Athena à partir de nombreux outils clients et applications SQL tiers. Vous configurez la connexion ODBC sur votre ordinateur client.

Considérations et restrictions

- Pour plus d'informations sur la migration du pilote ODBC Athena 1.x vers le pilote ODBC 2.x Athena, consultez Migration vers le pilote ODBC 2.x.
- Lorsque vous utilisez le récupérateur S3 avec l'option de chiffrement `CSE_KMS`, le client Amazon S3 ne peut pas déchiffrer le résultat stocké dans le compartiment Amazon S3. Pour contourner le problème, utilisez l'option d'API de streaming Athena pour récupérer le jeu de résultats.

Téléchargement du pilote ODBC 2.x

Pour télécharger le pilote ODBC Amazon Athena 2.x, consultez les liens figurant sur cette page.

Important

Lorsque vous utilisez le pilote ODBC 2.x, veillez à respecter les exigences suivantes :

- Ouvrez le port 444 – Conservez ouvert le port 444, utilisé par Athena pour diffuser les résultats de requête, pour le trafic sortant. Lorsque vous utilisez un PrivateLink point de terminaison pour vous connecter à Athena, assurez-vous que le groupe de sécurité attaché au PrivateLink point de terminaison est ouvert au trafic entrant sur le port 444.
- athena : GetQueryResultsStream policy — Ajoutez l'action `athena:GetQueryResultsStream` de stratégie aux principaux IAM qui utilisent le pilote ODBC. Cette action de politique n'est pas exposée directement avec l'API. Elle est uniquement utilisée avec les pilotes ODBC et JDBC dans le cadre de la prise en charge des résultats de streaming. Pour un exemple de politique, consultez AWS politique gérée : AWSQuicksightAthenaAccess.

Linux

Versions du pilote	Lien de téléchargement
ODBC 2.0.3.0 pour Linux 64 bits	Pilote ODBC Linux 64 bits 2.0.3.0 Pilote ODBC 2.0.3.0

macOS (ARM)

Versions du pilote	Lien de téléchargement
ODBC 2.0.3.0 pour macOS 64 bits (ARM)	Pilote ODBC pour macOS 64 bits 2.0.3.0 (ARM) Pilote ODBC pour macOS 2.0.3.0 (ARM)

macOS (Intel)

Versions du pilote	Lien de téléchargement
ODBC 2.0.3.0 pour macOS 64 bits (Intel)	Pilote ODBC pour macOS 64 bits 2.0.3.0 (Intel) Pilote ODBC pour macOS 2.0.3.0 (Intel)

Windows

Versions du pilote	Lien de téléchargement
ODBC 2.0.3.0 pour Windows 64 bits	Pilote ODBC pour Windows 64 bits 2.0.3.0 Pilote ODBC pour 2.0.3.0

Rubriques

- Notes de mise à jour Amazon Athena ODBC 2.x

Premiers pas avec le pilote ODBC 2.x

Utilisez les informations de cette section pour commencer à utiliser le pilote ODBC 2.x Amazon Athena. Le pilote est pris en charge sur les systèmes d'exploitation Windows, Linux et macOS.

Rubriques

- Windows
- Linux
- macOS

Windows

Si vous souhaitez utiliser un ordinateur client Windows pour accéder à Amazon Athena, le pilote ODBC Amazon Athena est requis.

Configuration système requise Windows

Installez le pilote ODBC Amazon Athena sur les ordinateurs clients qui accèderont directement aux bases de données Amazon Athena au lieu d'utiliser un navigateur Web.

Le système Windows que vous utilisez doit remplir les conditions suivantes :

- Vous avez des droits d'administrateur
- L'un des systèmes d'exploitation suivants :
  - Windows 11, 10 ou 8.1
  - Windows Server 2019, 2016 ou 2012
- Au moins 100 Mo d'espace disque disponible
- Microsoft Visual C++ Redistributable pour Visual Studio pour Windows 64 bits installé.

Installation du pilote ODBC Amazon Athena

Pour télécharger et installer le pilote ODBC Amazon Athena pour Windows

1. Téléchargez le fichier d'installation `AmazonAthenaODBC-2.x.x.x.msi`.
2. Lancez le fichier d'installation, puis choisissez Suivant.

3. Pour accepter les conditions du contrat de licence, cochez la case, puis choisissez Suivant.
4. Pour modifier l'emplacement d'installation, choisissez Parcourir, naviguez jusqu'au dossier souhaité, puis cliquez sur OK.
5. Pour accepter l'emplacement d'installation, choisissez Suivant.
6. Choisissez Installer.
7. Une fois l'installation terminée, choisissez Terminer.

Méthodes de définition des options de configuration du pilote

Pour contrôler le comportement du pilote ODBC Amazon Athena sous Windows, vous pouvez définir les options de configuration du pilote de la manière suivante :

- Dans le programme Administrateur de sources de données ODBC lorsque vous configurez un nom de source de données (DSN).
- En ajoutant ou en modifiant les clés de registre Windows à l'emplacement suivant :

```
HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\YOUR_DSN_NAME
```

- En définissant les options du pilote dans la chaîne de connexion lorsque vous vous connectez par programmation.

Configuration du nom d'une source de données sous Windows

Après avoir téléchargé et installé le pilote ODBC, vous devez ajouter une entrée de nom de source de données (DSN) à l'ordinateur client ou à l'instance Amazon EC2. Les outils clients SQL utilisent cette source de données pour se connecter à la base de données Amazon Athena et l'interroger.

Pour créer une entrée DSN système

1. Dans le menu Démarrer de Windows, cliquez avec le bouton droit sur Sources de données ODBC (64 bits), puis choisissez Plus, Exécuter en tant qu'administrateur.
2. Dans l'administrateur de sources de données ODBC, choisissez l'onglet Pilotes.
3. Dans la colonne Nom, vérifiez qu'Amazon Athena ODBC (x64) est présent.
4. Effectuez l’une des actions suivantes :

    - Pour configurer le pilote pour tous les utilisateurs de l'ordinateur, sélectionnez l'onglet DSN système. Les applications qui utilisent un compte différent pour charger les données peuvent

ne pas être en mesure de détecter les DSN utilisateur à partir d'un autre compte, nous recommandons l'option de configuration DSN système.

> **Note**
>
> L'utilisation de l'option DSN système nécessite des privilèges administratifs.

- Pour configurer le pilote uniquement pour votre compte utilisateur, choisissez l'onglet DSN utilisateur.

5. Choisissez Ajouter. La boîte de dialogue Créer une nouvelle source de données s'ouvre.
6. Choisissez Pilote ODBC Amazon Athena (x64), puis cliquez sur Terminer.
7. Dans la boîte de dialogue Configuration ODBC Amazon Athena, entrez les informations suivantes. Pour des informations détaillées sur ces options, consultez Principaux paramètres de connexion ODBC 2.x.

   - Dans Nom de la source de données, entrez le nom que vous souhaitez utiliser pour identifier la source de données.
   - Dans Description, entrez une description qui vous permettra d'identifier rapidement la source de données.
   - Dans Région, entrez le nom de la Région AWS dans laquelle vous utiliserez Athena (par exemple, **`us-west-1`**).
   - Dans Catalogue, entrez le nom du catalogue Amazon Athena. La valeur par défaut est AwsDataCatalog, qui est utilisée par AWS Glue.
   - Dans Base de données, entrez le nom de la base de données Amazon Athena. La valeur par défaut est Par défaut.
   - Dans Groupe de travail, entrez le nom du groupe de travail Amazon Athena. La valeur par défaut est primaire.
   - Pour l'emplacement de sortie S3, entrez l'emplacement dans Amazon S3 où les résultats de la requête seront stockés (par exemple,**`s3://DOC-EXAMPLE-BUCKET/`**).
   - (Facultatif) Dans Options de chiffrement, choisissez une option de chiffrement. L'argument par défaut est `NOT_SET`.
   - (Facultatif) Dans Clé KMS, choisissez une clé KMS de chiffrement si nécessaire.
8. Pour spécifier les options de configuration pour l'authentification IAM, choisissez Options d'authentification.
9. Entrez les informations suivantes :

- Dans Type d'authentification, choisissez Informations d'identification IAM. Il s'agit de l'option par défaut. Pour plus d'informations sur les types d'authentification disponibles, consultez Options d'authentification.
- Entrez un nom d'utilisateur dans Nom d'utilisateur.
- Dans Mot de passe, entrez un mot de passe.
- Pour le jeton de session, entrez un jeton de session si vous souhaitez utiliser des AWS informations d'identification temporaires. Pour plus d'informations sur les informations d'identification temporaires, consultez la section Utilisation des informations d'identification temporaires avec les AWS ressources du Guide de l'utilisateur IAM.

10. Choisissez OK.
11. Dans le bas de la boîte de dialogue Configuration ODBC Amazon Athena, choisissez Test. Si l'ordinateur client se connecte correctement à Amazon Athena, le champ Test de connexion indique Connexion réussie. Dans le cas contraire, la boîte indique Échec de la connexion avec les informations d'erreur correspondantes.
12. Pour fermer le test de connexion, cliquez sur OK. La source de données que vous avez créée apparaît désormais dans la liste des noms de source de données.

Utilisation d'une connexion sans DSN sous Windows

Vous pouvez utiliser une connexion sans DSN pour vous connecter à une base de données sans nom de source de données (DSN). L'exemple suivant montre une chaîne de connexion pour le pilote ODBC Amazon Athena (x64) qui se connecte à Amazon Athena.

```
DRIVER={Amazon Athena ODBC (x64)};Catalog=AwsDataCatalog;AwsRegion=us-
west-1;Schema=test_schema;S3OutputLocation=
s3://DOC-EXAMPLE-BUCKET/;AuthenticationType=IAM Credentials;UID=YOUR_UID;PWD=YOUR_PWD;
```

Linux

Si vous souhaitez utiliser un ordinateur client Linux pour accéder à Amazon Athena, le pilote ODBC Amazon Athena est requis.

Configuration système requise pour Linux

Chaque ordinateur client Linux sur lequel vous installez le pilote doit répondre aux exigences suivantes.

- Vous avez un accès root.
- Utilisez l'une des distributions Linux suivantes :
  - Red Hat Enterprise Linux (RHEL) 7 ou 8
  - CentOS 7 or 8.
- Disposer de 100 Mo d'espace disque disponible.
- Utilisez la version 2.3.1 ou ultérieure d'UnixODBC.
- Utilisez la version 2.26 ou ultérieure de la bibliothèque GNU C (glibc).

Installation du connecteur de données ODBC sous Linux

Suivez la procédure ci-dessous pour installer le pilote ODBC Amazon Athena sur un système d'exploitation Linux.

Pour installer le pilote ODBC Amazon Athena sous Linux

1. Entrez l'une des commandes suivantes :

   ```
   sudo rpm -Uvh AmazonAthenaODBC-2.X.Y.Z.rpm
   ```

   or

   ```
   sudo yum --nogpgcheck localinstall AmazonAthenaODBC-2.X.Y.Z.rpm
   ```

2. Une fois l'installation terminée, entrez l'une des commandes suivantes pour vérifier que le pilote est installé :

   - ```
     yum list | grep amazon-athena-odbc-driver
     ```

     Sortie :

     ```
     amazon-athena-odbc-driver.x86_64 2.0.2.1-1.amzn2int installed
     ```

   - ```
     rpm -qa | grep amazon
     ```

     Sortie :

```
amazon-athena-odbc-driver-2.0.2.1-1.amzn2int.x86_64
```

Configuration du nom d'une source de données sous Linux

Une fois le pilote installé, vous pouvez trouver des exemples `.odbc.ini` et `.odbcinst.ini` des fichiers à l'emplacement suivant :

- `/opt/athena/odbc/ini/`.

Utilisez les `.ini` fichiers de cet emplacement comme exemples pour configurer le pilote ODBC Amazon Athena et le nom de la source de données (DSN).

> **Note**
>
> Par défaut, les gestionnaires de pilotes ODBC utilisent les fichiers de configuration cachés `.odbc.ini` et `.odbcinst.ini` situés dans le répertoire de base.

Pour spécifier le chemin d'accès aux `.odbcinst.ini` fichiers `.odbc.ini` et à l'aide d'UnixODBC, effectuez les opérations suivantes.

Pour spécifier l'emplacement des **`.ini`** fichiers ODBC à l'aide d'UnixODBC

1. Définissez `ODBCINI` le chemin complet et le nom de fichier du `odbc.ini` fichier, comme dans l'exemple suivant.

   ```
   export ODBCINI=/opt/athena/odbc/ini/odbc.ini
   ```

2. Définissez `ODBCSYSINI` le chemin complet du répertoire contenant le `odbcinst.ini` fichier, comme dans l'exemple suivant.

   ```
   export ODBCSYSINI=/opt/athena/odbc/ini
   ```

3. Entrez la commande suivante pour vérifier que vous utilisez le gestionnaire de pilotes UnixODBC et les bons fichiers : `odbc*.ini`

   ```
   username % odbcinst -j
   ```

Exemple de sortie.

```
unixODBC 2.3.1
DRIVERS............: /opt/athena/odbc/ini/odbcinst.ini
SYSTEM DATA SOURCES: /opt/athena/odbc/ini/odbc.ini
FILE DATA SOURCES..: /opt/athena/odbc/ini/ODBCDataSources
USER DATA SOURCES..: /opt/athena/odbc/ini/odbc.ini
SQLULEN Size.......: 8
SQLLEN Size........: 8
SQLSETPOSIROW Size.: 8
```

4. Si vous souhaitez utiliser un nom de source de données (DSN) pour vous connecter à votre magasin de données, configurez le `odbc.ini` fichier pour définir les noms de source de données (DSN). Définissez les propriétés du `odbc.ini` fichier pour créer un DSN qui spécifie les informations de connexion pour votre magasin de données, comme dans l'exemple suivant.

```
[ODBC Data Sources]
athena_odbc_test=Amazon Athena ODBC (x64)

[ATHENA_WIDE_SETTINGS]  # Special DSN-name to signal driver about logging
 configuration.
LogLevel=0              # To enable ODBC driver logs, set this to 1.
UseAwsLogger=0          # To enable AWS-SDK logs, set this to 1.
LogPath=/opt/athena/odbc/logs/ # Path to store the log files. Permissions to the
 location are required.

[athena_odbc_test]
Driver=/opt/athena/odbc/lib/libathena-odbc.so
AwsRegion=us-west-1
Workgroup=primary
Catalog=AwsDataCatalog
Schema=default
AuthenticationType=IAM Credentials
UID=
PWD=
S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/
```

5. Configurez le `odbcinst.ini` fichier, comme dans l'exemple suivant.

```
[ODBC Drivers]
Amazon Athena ODBC (x64)=Installed
```

```
[Amazon Athena ODBC (x64)]
Driver=/opt/athena/odbc/lib/libathena-odbc.so
Setup=/opt/athena/odbc/lib/libathena-odbc.so
```

6. Après avoir installé et configuré le pilote ODBC Amazon Athena, utilisez l'outil de `isql` ligne de commande UnixODBC pour vérifier la connexion, comme dans l'exemple suivant.

```
username % isql -v "athena_odbc_test"
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

macOS

Si vous souhaitez utiliser un ordinateur client macOS pour accéder à Amazon Athena, le pilote ODBC Amazon Athena est requis.

Configuration système requise pour macOS

Chaque ordinateur macOS sur lequel vous installez le pilote doit répondre aux exigences suivantes.

- Utilisez macOS version 14 ou ultérieure.
- Disposer de 100 Mo d'espace disque disponible.
- Utilisez la version 3.52.16 ou ultérieure d'iODBC.

Installation du connecteur de données ODBC sur macOS

Suivez la procédure ci-dessous pour télécharger et installer le pilote ODBC Amazon Athena pour les systèmes d'exploitation macOS.

Pour télécharger et installer le pilote ODBC Amazon Athena pour macOS

1. Téléchargez le fichier `.pkg` du package.
2. Double-cliquez sur le fichier `.pkg`.

3. Suivez les étapes de l'assistant pour installer le pilote.
4. Sur la page du contrat de licence, appuyez sur Continuer, puis sélectionnez Accepter.
5. Choisissez Installer.
6. Une fois l'installation terminée, choisissez Terminer.
7. Entrez la commande suivante pour vérifier que le pilote est installé :

```
> pkgutil --pkgs | grep athenaodbc
```

En fonction de votre système, la sortie peut ressembler à l'une des suivantes.

```
com.amazon.athenaodbc-x86_64.Config
com.amazon.athenaodbc-x86_64.Driver
```

or

```
com.amazon.athenaodbc-arm64.Config
com.amazon.athenaodbc-arm64.Driver
```

Configuration du nom d'une source de données sous macOS

Une fois le pilote installé, vous pouvez trouver des exemples `.odbc.ini` et `.odbcinst.ini` des fichiers aux emplacements suivants :

- Ordinateurs à processeur Intel : `/opt/athena/odbc/x86_64/ini/`
- Ordinateurs à processeur ARM : `/opt/athena/odbc/arm64/ini/`

Utilisez les `.ini` fichiers de cet emplacement comme exemples pour configurer le pilote ODBC Amazon Athena et le nom de la source de données (DSN).

> **Note**
>
> Par défaut, les gestionnaires de pilotes ODBC utilisent les fichiers de configuration cachés `.odbc.ini` et `.odbcinst.ini` situés dans le répertoire de base.

Pour spécifier le chemin d'accès aux `.odbcinst.ini` fichiers `.odbc.ini` et à l'aide du gestionnaire de pilotes iODBC, effectuez les opérations suivantes.

Pour spécifier l'emplacement des **`.ini`** fichiers ODBC à l'aide du gestionnaire de pilotes IODBC

1. Définissez `ODBCINI` vers le chemin complet et le nom du fichier `odbc.ini`.

   - Pour les ordinateurs macOS équipés de processeurs Intel, utilisez la syntaxe suivante.

     ```
     export ODBCINI=/opt/athena/odbc/x86_64/ini/odbc.ini
     ```

   - Pour les ordinateurs macOS équipés de processeurs ARM, utilisez la syntaxe suivante.

     ```
     export ODBCINI=/opt/athena/odbc/arm64/ini/odbc.ini
     ```

2. Définissez `ODBCSYSINI` vers le chemin complet et le nom du fichier `odbcinst.ini`.

   - Pour les ordinateurs macOS équipés de processeurs Intel, utilisez la syntaxe suivante.

     ```
     export ODBCSYSINI=/opt/athena/odbc/x86_64/ini/odbcinst.ini
     ```

   - Pour les ordinateurs macOS équipés de processeurs ARM, utilisez la syntaxe suivante.

     ```
     export ODBCSYSINI=/opt/athena/odbc/arm64/ini/odbcinst.ini
     ```

3. Si vous souhaitez utiliser un nom de source de données (DSN) pour vous connecter à votre magasin de données, configurez le `odbc.ini` fichier pour définir les noms de source de données (DSN). Définissez les propriétés du `odbc.ini` fichier pour créer un DSN qui spécifie les informations de connexion pour votre magasin de données, comme dans l'exemple suivant.

   ```
   [ODBC Data Sources]
   athena_odbc_test=Amazon Athena ODBC (x64)

   [ATHENA_WIDE_SETTINGS] # Special DSN-name to signal driver about logging
    configuration.
   LogLevel=0             # set to 1 to enable ODBC driver logs
   UseAwsLogger=0         # set to 1 to enable AWS-SDK logs
   LogPath=/opt/athena/odbc/logs/ # Path to store the log files. Permissions to the
    location are required.

   [athena_odbc_test]
   Description=Amazon Athena ODBC (x64)
   ```

```
# For ARM:
Driver=/opt/athena/odbc/arm64/lib/libathena-odbc-arm64.dylib
# For Intel:
# Driver=/opt/athena/odbc/x86_64/lib/libathena-odbc-x86_64.dylib
AwsRegion=us-west-1
Workgroup=primary
Catalog=AwsDataCatalog
Schema=default
AuthenticationType=IAM Credentials
UID=
PWD=
S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/
```

4. Configurez le `odbcinst.ini` fichier, comme dans l'exemple suivant.

```
[ODBC Drivers]
Amazon Athena ODBC (x64)=Installed

[Amazon Athena ODBC (x64)]
# For ARM:
Driver=/opt/athena/odbc/arm64/lib/libathena-odbc-arm64.dylib
Setup=/opt/athena/odbc/arm64/lib/libathena-odbc-arm64.dylib
# For Intel:
# Driver=/opt/athena/odbc/x86_64/lib/libathena-odbc-x86_64.dylib
# Setup=/opt/athena/odbc/x86_64/lib/libathena-odbc-x86_64.dylib
```

5. Après avoir installé et configuré le pilote ODBC Amazon Athena, utilisez l'outil de `iodbctest` ligne de commande pour vérifier la connexion, comme dans l'exemple suivant.

```
username@ % iodbctest
iODBC Demonstration program
This program shows an interactive SQL processor
Driver Manager: 03.52.1623.0502

Enter ODBC connect string (? shows list): ?

DSN                             | Driver
-------------------------------------------------------------------------------
athena_odbc_test                | Amazon Athena ODBC (x64)

Enter ODBC connect string (? shows list): DSN=athena_odbc_test;
Driver: 2.0.2.1 (Amazon Athena ODBC Driver)
```

```
SQL>
```

Paramètres de connexion ODBC 2.x Athena

Les options de la boîte de dialogue Configuration ODBC Amazon Athena incluent les options d'authentification, les options avancées, les options de journalisation, les remplacements des points de terminaison et les options de proxy. Pour plus d'informations sur chacun d'entre eux, consultez les liens correspondants.

- Principaux paramètres de connexion ODBC 2.x
- Options d'authentification
- Options avancées
- Options de journalisation
- Remplacements des points de terminaison
- Options de proxy

Principaux paramètres de connexion ODBC 2.x

Les sections suivantes décrivent chacun des principaux paramètres de connexion.

Nom de la source de données

Spécifie le nom de la source de données.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
DSN	Facultatif pour les types de connexion sans DSN	`none`	`DSN=AmazonAthena0dbcUsWest1;`

Description

Contient la description de la source de données.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
Description	Facultatif	none	`Descripti on=Connection to Amazon Athena us-west-1;`

Catalogue

Spécifie le nom du catalogue de données. Pour plus d'informations sur les catalogues, consultez le DataCatalogmanuel Amazon Athena API Reference.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
Catalogue	Facultatif	`AwsDataCatalog`	`Catalog=A wsDataCatalog;`

Région

Spécifie le Région AWS. Pour plus d'informations Régions AWS, voir Régions et zones de disponibilité.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AwsRegion	Obligatoire	none	`AwsRegion =us-west-1;`

Base de données

Spécifie le nom de base de données. Pour plus d'informations sur les bases de données, consultez Base de données dans la Référence d'API Amazon Athena.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
Schema	Facultatif	`default`	`Schema=default;`

WorkGroup

Spécifie le nom du groupe de travail. Pour plus d'informations sur les groupes de travail, consultez le WorkGroupmanuel Amazon Athena API Reference.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
WorkGroup	Facultatif	`primary`	`Workgroup =primary;`

Emplacement de sortie

Spécifie l'emplacement dans Amazon S3 où les résultats de votre requête seront stockés. Pour plus d'informations sur l'emplacement de sortie, consultez ResultConfigurationle manuel Amazon Athena API Reference.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
S3 OutputLocation	Obligatoire	none	`S3OutputL ocation=s3:// DOC-EXAMPLE-BU CKET/;`

Options de chiffrement

Nom du paramètre de dialogue : options de chiffrement

Spécifie l'option de chiffrement. Pour plus d'informations sur les options de chiffrement, consultez EncryptionConfigurationle manuel Amazon Athena API Reference.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Valeurs possibles	Exemple de chaîne de connexion
S3 OutputEnc Option	Facultatif	none	`NOT_SET`, `SSE_S3`, `SSE_KMS`, `CSE_KMS`	`S3OutputEncOption=SSE_S3;`

Clé KMS

Spécifie une clé KMS pour le chiffrement. Pour plus d'informations sur la configuration du chiffrement des clés KMS, consultez EncryptionConfigurationle manuel Amazon Athena API Reference.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
Clé S3 OutputEnc KMS	Facultatif	none	`S3OutputEncKMSKey=your_key;`

Test de connexion

L'administrateur de sources de données ODBC propose une option Test que vous pouvez utiliser pour tester votre connexion ODBC 2.x à Amazon Athena. Pour les étapes, consultez Configuration du nom d'une source de données sous Windows. Lorsque vous testez une connexion, le pilote ODBC appelle l'action d'GetWorkGroupAPI Athena. L'appel utilise le type d'authentification et le fournisseur d'informations d'identification correspondant que vous avez spécifiés pour récupérer les informations d'identification. Le test de connexion est gratuit lorsque vous utilisez le pilote ODBC 2.x. Le test ne génère pas de résultats de requête dans votre compartiment Amazon S3.

Options d'authentification

Vous pouvez vous connecter à Amazon Athena à l'aide des types d'authentification suivants. Pour tous les types, le nom de la chaîne de connexion est `AuthenticationType`, le type de paramètre est `Required` et la valeur par défaut est `IAM Credentials`. Pour plus d'informations sur les

paramètres de chaque type d'authentification, rendez-vous sur le lien correspondant. Pour les paramètres d'authentification courants, consultez Paramètres d'authentification communs.

Type d'authentification	Exemple de chaîne de connexion
Informations d'identification IAM	`AuthenticationType=IAM Credentials;`
Profil IAM	`AuthenticationType=IAM Profile;`
AD FS	`AuthenticationType=ADFS;`
Azure AD	`AuthenticationType=AzureAD;`
Navigateur Azure AD	`AuthenticationType=BrowserAzureAD;`
Navigateur SAML	`AuthenticationType=BrowserSAML;`
Navigateur SSO OIDC	`AuthenticationType=BrowserSSOOIDC;`
Informations d'identification par défaut	`AuthenticationType=Default Credentials;`
External Credentials	`AuthenticationType=External Credentials;`
Profil d'instance	`AuthenticationType=Instance Profile;`
JWT	`AuthenticationType=JWT;`
Okta	`AuthenticationType=Okta;`
Ping	`AuthenticationType=Ping;`

Informations d'identification IAM

Vous pouvez utiliser vos informations d'identification IAM pour vous connecter à Amazon Athena avec le pilote ODBC à l'aide des paramètres de chaîne de connexion décrits dans cette section.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=IAM Credentials;`

ID de l'utilisateur

L'identifiant de votre clé d' AWS accès. Pour plus d'informations sur les clés d'accès, consultez Informations d'identification de sécuritéAWS dans le Guide de l'utilisateur IAM.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UID	Obligatoire	`none`	`UID=AKIAIOSFODNN7EXAMPLE;`

Mot de passe

L'identifiant de votre clé AWS secrète. Pour plus d'informations sur les clés d'accès, consultez Informations d'identification de sécuritéAWS dans le Guide de l'utilisateur IAM.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
PWD	Obligatoire	`none`	`PWD=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKE;`

Jeton de session

Si vous utilisez des AWS informations d'identification temporaires, vous devez spécifier un jeton de session. Pour plus d'informations sur les informations d'identification temporaires, consultez Informations d'identification de sécurité temporaires dans le Guide de l'utilisateur IAM.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
SessionToken	Facultatif	none	`SessionToken=AQoDYXdzEJr...<remainder of session token>;`

Profil IAM

Vous pouvez configurer un profil nommé pour vous connecter à Amazon Athena à l'aide du pilote ODBC. Pour utiliser les informations d'identification disponibles dans votre profil d'instance Amazon EC2 d'hébergement, définissez le paramètre `credential_source` sur `Ec2InstanceMetadata`. Si vous souhaitez utiliser un fournisseur d'informations d'identification personnalisé dans un profil nommé, spécifiez une valeur pour le paramètre `plugin_name` dans la configuration de votre profil.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=IAM Profile;`

Profil AWS

Le nom de profil à utiliser pour votre connexion ODBC. Pour plus d'informations sur les profils, consultez Utilisation des profils nommés dans le Guide de l'utilisateur d'AWS Command Line Interface.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AWSProfile	Obligatoire	none	`AWSProfile=default;`

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Le paramètre de rôle préféré est utilisé lorsque le fournisseur d'informations d'identification personnalisé est spécifié par le paramètre `plugin_name` dans la configuration de votre profil. Pour plus d'informations sur les rôles ARN, consultez AssumeRole dans la Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
preferred_role	Facultatif	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durée de la session

La durée de la session de rôle en secondes. Pour plus d'informations sur la durée de la session, consultez AssumeRole dans le document Référence d'API AWS Security Token Service. Le paramètre de durée de la session est utilisé lorsque le fournisseur d'informations d'identification personnalisé est spécifié par le paramètre `plugin_name` dans la configuration de votre profil.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
duration	Facultatif	`900`	`duration=900;`

Nom du plug-in

Spécifie le nom d'un fournisseur d'informations d'identification personnalisé utilisé dans un profil nommé. Ce paramètre peut prendre les mêmes valeurs que celles du champ Type d'authentification de l'administrateur de sources de données ODBC, mais il n'est utilisé que par la configuration `AWSProfile`.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
plugin_name	Facultatif	`none`	`plugin_name=AzureAD;`

AD FS

AD FS est un plug-in d'authentification basé sur SAML qui fonctionne avec le fournisseur d'identité Active Directory Federation Service (AD FS). Le plug-in prend en charge l'authentification Windows intégrée et l'authentification par formulaire. Si vous utilisez l'authentification Windows intégrée, vous pouvez omettre le nom d'utilisateur et le mot de passe. Pour plus d'informations sur la configuration d'AD FS et d'Athena, consultez Configuration de l'accès fédéré à Amazon Athena pour les utilisateurs de Microsoft AD FS à l'aide d'un client ODBC.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=ADFS;`

ID de l'utilisateur

Votre nom d'utilisateur pour vous connecter au serveur AD FS. Pour l'authentification Windows intégrée, vous pouvez omettre le nom d'utilisateur. Si votre configuration AD FS nécessite un nom d'utilisateur, vous devez le fournir dans le paramètre de connexion.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UID	Facultatif pour l'authentification Windows intégrée	none	`UID=domain\username;`

Mot de passe

Votre mot de passe pour vous connecter au serveur AD FS. Tout comme le champ du nom d'utilisateur, vous pouvez omettre le mot de passe si vous utilisez l'authentification Windows intégrée. Si votre configuration AD FS nécessite un mot de passe, vous devez le fournir dans le paramètre de connexion.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
PWD	Facultatif pour l'authentification Windows intégrée	none	`PWD=password_3EXAMPLE;`

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Si votre assertion SAML comporte plusieurs rôles, vous pouvez spécifier ce paramètre pour choisir le rôle à assumer. Ce rôle doit être présent dans l'assertion SAML. Pour plus d'informations sur les rôles ARN, consultez AssumeRole dans la Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
preferred_role	Facultatif	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durée de la session

La durée de la session de rôle en secondes. Pour plus d'informations sur la durée de la session, consultez AssumeRole dans la Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
duration	Facultatif	`900`	`duration=900;`

Hôte IdP

Nom de l'hôte de service AD FS.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
idp_host	Require	`none`	`idp_host=<server-name>.<company.com>;`

Port IdP

Port à utiliser pour se connecter à l'hôte AD FS.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
idp_port	Obligatoire	`none`	`idp_port=443;`

LoginToRP

Partie utilisatrice approuvée. Utilisez ce paramètre pour remplacer l'URL du point de terminaison de la partie utilisatrice AD FS.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
LoginToRP	Facultatif	`urn:amazon:webservices`	`LoginToRP=trustedparty;`

Azure AD

Azure AD est un plug-in d'authentification basé sur SAML qui fonctionne avec le fournisseur d'identité Azure AD. Ce plug-in ne prend pas en charge l'authentification multifactorielle (MFA). Si vous avez besoin d'un support MFA, pensez plutôt à utiliser le plug-in `BrowserAzureAD`.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=AzureAD;`

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Pour plus d'informations sur les rôles ARN, consultez AssumeRole dans la Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
preferred_role	Facultatif	none	`preferred_role=arn:aws:iam::123456789012:id/user1;`

Durée de la session

La durée de la session de rôle en secondes. Pour plus d'informations, consultez AssumeRole dans la Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
duration	Facultatif	`900`	`duration=900;`

ID de locataire

Spécifie l'ID de locataire de votre application.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
idp_tenant	Obligatoire	none	`idp_tenant=123zz112z-z12d-1z1f-11zz-f111aa111234;`

ID de client

Spécifie l'ID de client de votre application.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
client_id	Obligatoire	none	`client_id=9178ac27-a1bc-1a2b-1a2b-a123abcd1234;`

Secret client

Spécifie le secret de votre client.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
client_secret	Obligatoire	none	`client_secret=zG12q~.xzG1xx`

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
			`xZ1wX1.~Z zXXX1XxkH ZizeT1zzZ;`

Navigateur Azure AD

Le navigateur Azure AD est un plug-in d'authentification basé sur SAML qui fonctionne avec le fournisseur d'identité Azure AD et prend en charge l'authentification multifactorielle. Contrairement au plug-in Azure AD standard, celui-ci ne nécessite pas de nom d'utilisateur, de mot de passe ou de secret client dans les paramètres de connexion.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=BrowserAzureAD;`

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Si votre assertion SAML comporte plusieurs rôles, vous pouvez spécifier ce paramètre pour choisir le rôle à assumer. Le rôle spécifié doit être présent dans l'assertion SAML. Pour plus d'informations sur les rôles ARN, consultez AssumeRole dans la Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
preferred_role	Facultatif	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durée de la session

La durée de la session de rôle en secondes. Pour plus d'informations sur la durée de la session, consultez AssumeRole dans le document Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
duration	Facultatif	`900`	`duration=900;`

ID de locataire

Spécifie l'ID de locataire de votre application.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
idp_tenant	Obligatoire	`none`	`idp_tenant=123zz112z-z12d-1z1f-11zz-f111aa111234;`

ID de client

Spécifie l'ID de client de votre application.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
client_id	Obligatoire	`none`	`client_id=9178ac27-a1bc-1a2b-1a2b-a123abcd1234;`

Expiration

Durée, en secondes, avant que le plug-in arrête d'attendre la réponse SAML d'Azure AD.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
timeout	Facultatif	`120`	`timeout=90;`

Activation du cache de fichiers Azure

Active un cache d'informations d'identification temporaire. Ce paramètre de connexion permet de mettre en cache des informations d'identification temporaires et de les réutiliser entre plusieurs processus. Utilisez cette option pour réduire le nombre de fenêtres de navigateur ouvertes lorsque vous utilisez des outils de BI tels que Microsoft Power BI.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
browser_azure_cache	Facultatif	1	`browser_azure_cache=0;`

Navigateur SAML

Le navigateur SAML est un plug-in d'authentification générique qui peut fonctionner avec les fournisseurs d'identité basés sur SAML et prend en charge l'authentification multifactorielle. Pour des informations de configuration détaillées, consultez Configuration de l'authentification unique à l'aide d'ODBC, SAML 2.0 et du fournisseur d'identité Okta.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=BrowserSAML;`

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Si votre assertion SAML comporte plusieurs rôles, vous pouvez spécifier ce paramètre pour choisir le rôle à assumer. Ce rôle doit être présent dans l'assertion SAML. Pour plus d'informations sur les rôles ARN, consultez AssumeRole dans la Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
preferred_role	Facultatif	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durée de la session

La durée de la session de rôle en secondes. Pour plus d'informations, consultez AssumeRole dans la Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
duration	Facultatif	`900`	`duration=900;`

URL de connexion

URL d'authentification unique qui s'affiche pour votre application.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
login_url	Obligatoire	none	`login_url=https://trial-1234567.okta.com/app/trial-123`

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
			`4567_okta browsersaml_1/ zzz4izzzAzDFB zZz1234/sso/ saml;`

Port d'écoute

Numéro de port utilisé pour écouter la réponse SAML. Cette valeur doit correspondre à l'URL d'IAM Identity Center avec laquelle vous avez configuré l'IdP (par exemple, `http://localhost:7890/athena`).

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
listen_port	Facultatif	`7890`	`listen_po rt=7890;`

Expiration

Durée, en secondes, avant que le plug-in arrête d'attendre la réponse SAML du fournisseur d'identité.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
timeout	Facultatif	`120`	`timeout=90;`

Navigateur SSO OIDC

Browser SSO OIDC est un plugin d'authentification qui fonctionne avec. AWS IAM Identity Center Pour plus d'informations sur l'activation et l'utilisation d'IAM Identity Center, consultez Étape 1 : Activer IAM Identity Center dans le Guide de l'utilisateur AWS IAM Identity Center .

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=BrowserSSOOIDC;`

URL de démarrage d'IAM Identity Center

URL du portail AWS d'accès. L'action d'StartDeviceAuthorizationAPI IAM Identity Center utilise cette valeur pour le `startUrl` paramètre.

Pour copier l'URL du portail AWS d'accès

1. Connectez-vous à la AWS IAM Identity Center console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/singlesignon/.
2. Dans le panneau de navigation, sélectionnez Settings (Paramètres).
3. Sur la page Paramètres, sous Source d'identité, choisissez l'icône du presse-papier pour l'URL du portail d'accès AWS .

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
sso_oidc_start_url	Obligatoire	none	`sso_oidc_start_url=https://app_id.awsapps.com/start;`

Région du centre d'identité IAM

L' Région AWS endroit où votre SSO est configuré. Les clients du `SSOClient` AWS SDK `SSOOIDCClient` et utilisent cette valeur pour le `region` paramètre.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
sso_oidc_region	Obligatoire	none	`sso_oidc_region=us-east-1;`

Scopes

Liste des portées définies par le client. Au moment de l'autorisation, cette liste restreint les autorisations lorsqu'un jeton d'accès est accordé. L'action d'RegisterClientAPI IAM Identity Center utilise cette valeur pour le `scopes` paramètre.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
sso_oidc_scopes	Facultatif	none	`sso_oidc_scopes=scope1,scope2,scope3;`

ID de compte

L'identifiant attribué à l'utilisateur. Compte AWS L'GetRoleCredentialsAPI IAM Identity Center utilise cette valeur pour le `accountId` paramètre.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
sso_oidc_account_id	Obligatoire	none	`sso_oidc_account_id=123456789123;`

Nom du rôle

Nom descriptif du rôle attribué à l'utilisateur. Le nom que vous spécifiez pour cet ensemble d'autorisations apparaît dans le portail AWS d'accès en tant que rôle disponible. L'action d'GetRoleCredentialsAPI IAM Identity Center utilise cette valeur pour le `roleName` paramètre.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
sso_oidc_role_name	Obligatoire	none	`sso_oidc_role_name=AthenaReadAccess;`

Expiration

Nombre de secondes pendant lesquelles l'API SSO d'interrogation doit vérifier la présence du jeton d'accès.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
sso_oidc_timeout	Facultatif	120	`sso_oidc_timeout=60;`

Activation du cache de fichiers

Active un cache d'informations d'identification temporaire. Ce paramètre de connexion permet de mettre en cache des informations d'identification temporaires et de les réutiliser entre plusieurs processus. Utilisez cette option pour réduire le nombre de fenêtres de navigateur ouvertes lorsque vous utilisez des outils de BI tels que Microsoft Power BI.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
sso_oidc_cache	Facultatif	1	`sso_oidc_cache=0;`

Informations d'identification par défaut

Vous pouvez utiliser les informations d'identification par défaut que vous configurez sur votre système client pour vous connecter à Amazon Athena. Pour plus d'informations sur l'utilisation des informations d'identification par défaut, consultez Utilisation de la chaîne de fournisseurs d'informations d'identification par défaut dans le Guide du développeur AWS SDK for Java.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=Default Credentials;`

External Credentials

External Credentials est un plug-in que vous pouvez utiliser pour vous connecter à n'importe quel fournisseur d'identité externe basé sur SAML. Pour utiliser le plug-in, vous devez transmettre un fichier exécutable qui renvoie une réponse SAML.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=External Credentials;`

Chemin exécutable

Le chemin d'accès à l'exécutable qui suit la logique de votre fournisseur d'informations d'identification personnalisé basé sur SAML. La sortie de l'exécutable doit être la réponse SAML analysée du fournisseur d'identité.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
ExecutablePath	Obligatoire	none	ExecutablePath=C:\Users *user_name* *external_ credential.exe*

Liste d'arguments

La liste des arguments que vous souhaitez transmettre à l'exécutable.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
ArgumentList	Facultatif	none	ArgumentList= *arg1 arg2 arg3*

Profil d'instance

Ce type d'authentification est utilisé sur les instances EC2 et est fourni via le service de métadonnées Amazon EC2.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	IAM Credentia ls	AuthenticationType =Instance Profile;

JWT

Le plug-in JWT (JSON Web Token) fournit une interface qui utilise des jetons Web JSON pour assumer un rôle Amazon IAM. La configuration dépend du fournisseur d'identité. Pour

plus d'informations sur la configuration de la fédération pour Google Cloud et AWS, consultez Configuration de la fédération d'identité de charge de travail avec AWS ou Azure dans la documentation Google Cloud.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=JWT;`

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Pour plus d'informations sur les rôles ARN, consultez AssumeRole dans la Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
preferred_role	Facultatif	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durée de la session

La durée de la session de rôle en secondes. Pour plus d'informations sur la durée de la session, consultez AssumeRole dans le document Référence d'API AWS Security Token Service.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
duration	Facultatif	`900`	`duration=900;`

Jeton Web JSON

Le jeton Web JSON utilisé pour récupérer les informations d'identification temporaires IAM à l'aide de l'action d'API AWS STS AssumeRoleWithWebIdentity. Pour plus d'informations sur la génération de jetons Web JSON pour les utilisateurs de Google Cloud Platform (GCP), consultez Utilisation des jetons JWT OAuth dans la documentation Google Cloud.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
`web_identity_token`	Obligatoire	`none`	`web_identity_token=eyJhbGc...<remainder of token>;`

Nom de la session de rôle

Le nom de la session. Une technique courante consiste à utiliser le nom ou l'identifiant de l'utilisateur de votre application comme nom de session du rôle. Les informations d'identification de sécurité temporaires utilisées par votre application sont ainsi associées de manière pratique à l'utilisateur correspondant.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
role_session_name	Obligatoire	`none`	`role_session_name=familiarname;`

Okta

Okta est un plug-in d'authentification basé sur SAML qui fonctionne avec le fournisseur d'identité Okta. Pour plus d'informations sur la configuration de la fédération pour Okta et Amazon Athena, consultez Configuration de SSO pour ODBC en utilisant le plugin Okta et le fournisseur d'identité Okta.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=Okta;`

ID de l'utilisateur

Votre nom d'utilisateur Okta.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UID	Obligatoire	none	`UID=jane.doe@org.com;`

Mot de passe

Votre mot de passe d'utilisateur Okta.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
PWD	Obligatoire	none	`PWD=oktauserpasswordexample;`

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Pour plus d'informations sur les rôles ARN, consultez AssumeRolela référence de l'AWS Security Token Service API.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
preferred_role	Facultatif	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durée de la session

La durée de la session de rôle en secondes. Pour plus d'informations, consultez AssumeRolela référence de AWS Security Token Service l'API.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
duration	Facultatif	`900`	`duration=900;`

Hôte IdP

URL de votre organisation Okta. Vous pouvez extraire le paramètre `idp_host` de l'URL de lien intégré dans votre application Okta. Pour les étapes, consultez Récupération des informations de configuration ODBC d'Okta. Le premier segment suivant `https://`, jusqu'à `okta.com` inclus, est votre hôte IdP (par exemple, `http://trial-1234567.okta.com`).

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
idp_host	Obligatoire	`None`	`idp_host=dev-99999999.okta.com;`

Port IdP

Le numéro de port à utiliser pour vous connecter à votre hôte IdP.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
idp_port	Obligatoire	None	idp_port=443;

ID de l'application Okta

Saisissez l'identifiant en deux parties de votre application. Vous pouvez extraire le paramètre app_id de l'URL de lien intégré dans votre application Okta. Pour les étapes, consultez Récupération des informations de configuration ODBC d'Okta. L'ID de l'application correspond aux deux derniers segments de l'URL, y compris la barre oblique au milieu. Les segments sont deux chaînes de 20 caractères contenant un mélange de chiffres et de lettres majuscules et minuscules (par exemple, Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4).

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
app_id	Obligatoire	None	app_id=0oa25kx8ze9A3example/alnexamplea0piaWa0g7;

Nom de l'application Okta

Le nom de l'application Okta.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
app_name	Obligatoire	None	app_name=amazon_aws_redshift;

Temps d'attente Okta

Spécifie la durée en secondes d'attente du code d'authentification multifactorielle (MFA).

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
okta_mfa_wait_time	Facultatif	`10`	`okta_mfa_wait_time=20;`

Type MFA Okta

Le type de facteur MFA. Les types pris en charge sont Google Authenticator, SMS (Okta), Okta Verify avec Push et Okta Verify avec TOTP. Les politiques de sécurité de chaque organisation déterminent si l'authentification multifactorielle est requise pour la connexion des utilisateurs.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Valeurs possibles	Exemple de chaîne de connexion
okta_mfa_type	`Optional`	`None`	`googleauthenticator, smsauthentication, oktaverifywithpush, oktaverifywithtotp`	`okta_mfa_type=oktaverifywithpush;`

Numéro de téléphone Okta

Numéro de téléphone à utiliser pour l' AWS SMS authentification. Ce paramètre est requis uniquement pour l'inscription multifactorielle. Si votre numéro de téléphone portable est déjà inscrit, ou si l'authentification AWS SMS n'est pas utilisée par la politique de sécurité, vous pouvez ignorer ce champ.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
okta_mfa_phone_number	Obligatoire pour l'inscription à la MFA, facultatif dans le cas contraire	None	`okta_mfa_phone_number=19991234567;`

Activation du cache de fichiers Okta

Active un cache d'informations d'identification temporaire. Ce paramètre de connexion permet de mettre en cache des informations d'identification temporaires et de les réutiliser entre les multiples processus ouverts par les applications BI. Utilisez cette option pour éviter la limitation de l'API Okta.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
okta_cache	Facultatif	`0`	`okta_cache=1;`

Ping

Ping est un plugin basé sur SAML qui fonctionne avec le fournisseur PingFederated d'identité.

Type d'authentification

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
AuthenticationType	Obligatoire	`IAM Credentials`	`AuthenticationType=Ping;`

ID de l'utilisateur

Le nom d'utilisateur du PingFederate serveur.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UID	Obligatoire	none	`UID=pingusername@domain.com;`

Mot de passe

Le mot de passe du PingFederate serveur.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
PWD	Obligatoire	none	`PWD=pingpassword;`

Rôle préféré

Amazon Resource Name (ARN) du rôle à assumer. Si votre assertion SAML comporte plusieurs rôles, vous pouvez spécifier ce paramètre pour choisir le rôle à assumer. Ce rôle doit être présent dans l'assertion SAML. Pour plus d'informations sur les rôles ARN, consultez AssumeRolela référence de l'AWS Security Token Service API.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
preferred_role	Facultatif	none	`preferred_role=arn:aws:iam::123456789012:id/user1;`

Durée de la session

La durée de la session de rôle en secondes. Pour plus d'informations sur la durée de session, consultez AssumeRolela référence de l'AWS Security Token Service API.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
duration	Facultatif	`900`	`duration=900;`

Hôte IdP

L'adresse de votre serveur Ping. Pour trouver votre adresse, rendez-vous sur l'URL suivante et consultez le champ Point de terminaison de l'application SSO.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
idp_host	Obligatoire	`none`	`idp_host=ec2-1-83-65-12.compute-1.amazonaws.com;`

Port IdP

Le numéro de port à utiliser pour vous connecter à votre hôte IdP.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
idp_port	Obligatoire	`None`	`idp_port=443;`

Partenaire SPID

Adresse du fournisseur de services. Pour trouver l'adresse du fournisseur de services, rendez-vous sur l'URL suivante et consultez le champ Point de terminaison de l'application SSO.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
partner_spid	Obligatoire	None	`partner_spid=https://us-east-1.signin.aws.amazon.com/platform/saml/<...>;`

Paramètre URI Ping

Transmet à Ping un argument URI pour une demande d'authentification. Utilisez ce paramètre pour contourner la limitation du rôle unique de Lake Formation. Configurez Ping pour reconnaître le paramètre transmis et vérifiez que le rôle transmis existe dans la liste des rôles assignés à l'utilisateur. Envoyez ensuite un rôle unique dans l'assertion SAML.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
ping_uri_param	Facultatif	None	`ping_uri_param=role=my_iam_role;`

Paramètres d'authentification communs

Les paramètres de cette section sont communs aux types d'authentification, comme indiqué.

Utiliser un proxy pour IdP

Permet la communication entre le pilote et l'IdP via le proxy. Cette option est disponible pour les plug-ins d'authentification suivants :

- AD FS
- Azure AD
- Navigateur Azure AD
- Navigateur SSO OIDC
- JWT

- Okta
- Ping

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UseProxyForIdP	Facultatif	`0`	`UseProxyForIdP=1;`

Utiliser Lake Formation

Utilise l'action d'API AssumeDecoratedRoleWithSAML de Lake Formation pour récupérer des informations d'identification IAM temporaires au lieu de l'action d'API AssumeRoleWithSAML de AWS STS. Cette option est disponible pour les plug-ins d'authentification d'Azure AD, du navigateur Azure AD, du navigateur SAML, d'Okta, de Ping et d'AD FS.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
LakeformationEnabled	Facultatif	`0`	`LakeformationEnabled=1;`

SSL non sécurisé (IdP)

Désactive SSL lors de la communication avec l'IdP. Cette option est disponible pour les plug-ins d'authentification d'Azure AD, du navigateur Azure AD, d'Okta, de Ping et d'AD FS.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
SSL_Insecure	Facultatif	`0`	`SSL_Insecure=1;`

Remplacements des points de terminaison

Remplacement du point de terminaison Athena

La classe `endpointOverride ClientConfiguration` utilise cette valeur pour remplacer le point de terminaison HTTP par défaut pour le client Amazon Athena. Pour plus d'informations, consultez Configuration client AWS du Guide du développeur AWS SDK for C++ .

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
EndpointOverride	Facultatif	none	`EndpointOverride=athena.us-west-2.amazonaws.com;`

Remplacement du point de terminaison de streaming Athena

La méthode `ClientConfiguration.endpointOverride` utilise cette valeur pour remplacer le point de terminaison HTTP par défaut pour le client de streaming Amazon Athena. Pour plus d'informations, consultez Configuration client AWS dans le Guide du développeur AWS SDK for C++ . Le service Athena Streaming est disponible via le port 444.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
Streaming EndpointOverride	Facultatif	none	`StreamingEndpointOverride=athena.us-west-1.amazonaws.com:444;`

AWS STS remplacement du point de terminaison

La `ClientConfiguration.endpointOverride` méthode utilise cette valeur pour remplacer le point de terminaison HTTP par défaut pour le AWS STS client. Pour plus d'informations, consultez Configuration client AWS du Guide du développeur AWS SDK for C++ .

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
StsEndpointOverride	Facultatif	none	`StsEndpointOverride=sts.us-west-1.amazonaws.com;`

Remplacement des point de terminaison Lake Formation

La méthode `ClientConfiguration.endpointOverride` utilise cette valeur pour remplacer le point de terminaison HTTP par défaut pour le client Lake Formation. Pour plus d'informations, consultez Configuration client AWS du Guide du développeur AWS SDK for C++ .

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
LakeFormationEndpointOverride	Facultatif	none	`LakeFormationEndpointOverride=lakeformation.us-west-1.amazonaws.com;`

Remplacement des points de terminaison SSO

La méthode `ClientConfiguration.endpointOverride` utilise cette valeur pour remplacer le point de terminaison HTTP par défaut pour le client SSO. Pour plus d'informations, consultez Configuration client AWS du Guide du développeur AWS SDK for C++ .

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
SSO EndpointOverride	Facultatif	none	`SSOEndpointOverride=portal.sso.us-east-2.amazonaws.com;`

Remplacement du point de terminaison SSO OIDC

La méthode `ClientConfiguration.endpointOverride` utilise cette valeur pour remplacer le point de terminaison HTTP par défaut pour le client SSO OIDC. Pour plus d'informations, consultez Configuration client AWS du Guide du développeur AWS SDK for C++ .

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
SSOÏDC EndpointOverride	Facultatif	`none`	`SSOOIDCEndpointOverride=oidc.us-east-2.amazonaws.com`

Options avancées

Taille d'extraction

Nombre maximal de résultats (lignes) à renvoyer dans cette demande. Pour plus d'informations sur les paramètres, consultez GetQuery MaxResults. Pour l'API de streaming, la valeur maximale est 10000000.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
RowsToFetchPerBlock	Facultatif	`1000` pour les applications autres que le streaming `20000` pour le streaming	`RowsToFetchPerBlock=20000;`

Activer la réutilisation des résultats

Spécifie si les résultats de requête précédents peuvent être réutilisés lors de l'exécution de la requête. Pour plus d'informations sur les paramètres, consultez ResultReuseByAgeConfiguration.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
EnableResultReuse	Facultatif	`0`	`EnableResultReuse=1;`

Âge maximum de réutilisation des résultats

Spécifie, en minutes, l'âge maximum d'un résultat de requête précédent qu'Athena doit envisager de réutiliser. Pour plus d'informations sur les paramètres, consultez ResultReuseByAgeConfiguration.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
ReusedResultMaxAgeInMinutes	Facultatif	`60`	`ReusedResultMaxAgeInMinutes=90;`

Activer l'API de streaming

Détermine s'il faut utiliser l'API de streaming Athena pour extraire le jeu de résultats.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UseResultsetStreaming	Facultatif	`0`	`UseResultsetStreaming=1;`

Activer le récupérateur S3

Récupère l'ensemble de résultats généré par Athena depuis le compartiment Amazon S3 en interagissant directement avec Amazon S3.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
EnableS3Fetcher	Facultatif	1	`EnableS3Fetcher=1;`

Utiliser plusieurs threads S3

Récupère les données depuis Amazon S3 à l'aide de plusieurs threads. Lorsque cette option est activée, le fichier de résultat stocké dans le compartiment Amazon S3 est extrait en parallèle à l'aide de plusieurs threads.

Activez cette option uniquement si vous disposez d'une bonne bande passante réseau. Par exemple, lors de nos mesures sur une instance EC2 c5.2xlarge, un client S3 à thread unique a atteint 1 Gbit/s, tandis que les clients S3 à plusieurs threads ont atteint 4 Gbit/s de débit réseau.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UseMultipleFils S3	Facultatif	`0`	`UseMultipleS3Threads=1;`

Utiliser un catalogue et un schéma uniques

Par défaut, le pilote ODBC interroge Athena pour obtenir la liste des catalogues et schémas disponibles. Cette option oblige le pilote à utiliser le catalogue et le schéma spécifiés par la boîte de dialogue de configuration de l'administrateur de sources de données ODBC ou par les paramètres de connexion.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UseSingleCatalogAndSchema	Facultatif	`0`	`UseSingleCatalogAndSchema=1;`

Utiliser la requête pour répertorier les tables

Pour les types de `LAMBDA` catalogue, permet au pilote ODBC de soumettre une SHOW TABLES requête pour obtenir la liste des tables disponibles. Il s'agit de la valeur par défaut. Si ce paramètre est défini sur 0, le pilote ODBC utilise l'API ListTableMetadataAthena pour obtenir la liste des tables disponibles. Notez que, pour les types de `LAMBDA` catalogue, l'utilisation `ListTableMetadata` entraîne une régression des performances.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UseQueryToListTabl es	Facultatif	`1`	`UseQueryToListTabl es=1;`

Utiliser WCHAR pour les types de chaînes

Par défaut, le pilote ODBC utilise `SQL_CHAR` et `SQL_VARCHAR` pour Athena les `char` types de données de chaîne`varchar`,,,`string`, `array` `map<>struct<>`, et. `row` La définition de ce paramètre 1 force le pilote à utiliser `SQL_WCHAR` et `SQL_WVARCHAR` pour les types de données de chaîne. Les types de caractères larges et de caractères variables larges sont utilisés pour garantir que les caractères de différentes langues peuvent être stockés et récupérés correctement.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
Utiliser W CharForSt ringTypes	Facultatif	`0`	`UseWCharForStringT ypes=1;`

Interroger des catalogues externes

Spécifie si le pilote doit interroger des catalogues externes à partir d'Athena. Pour plus d’informations, consultez Migration vers le pilote ODBC 2.x.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
QueryExternalCatalogs	Facultatif	`0`	`QueryExternalCatalogs=1;`

Vérifier le certificat SSL

Contrôle s'il convient de vérifier les certificats SSL lorsque vous utilisez le AWS SDK. Cette valeur est transmise au paramètre `ClientConfiguration.verifySSL`. Pour plus d'informations, consultez Configuration client AWS du Guide du développeur AWS SDK for C++ .

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
VerifySSL	Facultatif	`1`	`VerifySSL=0;`

Taille du bloc de résultats S3

Spécifie, en octets, la taille du bloc à télécharger pour une seule demande d'GetObjectAPI Amazon S3. La valeur par défaut est de 67108864 (64 Mo). Les valeurs minimale et maximale autorisées sont 10485760 (10 Mo) et 2146435072 (environ 2 Go).

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
S3 ResultBlockSize	Facultatif	`67108864`	`S3ResultBlockSize=268435456;`

Longueur de colonne de chaîne

Spécifie la longueur des colonnes contenant le type de `string` données. Comme Athena utilise le type de données de chaîne Apache Hive, dont la précision n'est pas définie, la longueur par défaut signalée par Athena est 2147483647 (). `INT_MAX` Comme les outils de BI préallouent généralement

de la mémoire aux colonnes, cela peut entraîner une consommation de mémoire élevée. Pour éviter cela, le pilote ODBC Athena limite la précision signalée pour les colonnes du type de `string` données et expose le paramètre de `StringColumnLength` connexion afin que la valeur par défaut puisse être modifiée.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
StringColumnLength	Facultatif	255	`StringColumnLength =65535;`

Longueur de colonne de type complexe

Spécifie la longueur des colonnes contenant des types de données complexes tels que `mapstruct`, et`array`. Par exemple StringColumnLength, Athena indique une précision nulle pour les colonnes contenant des types de données complexes. Le pilote ODBC Athena définit la précision par défaut pour les colonnes contenant des types de données complexes et expose le paramètre de `ComplexTypeColumnLength` connexion afin que la valeur par défaut puisse être modifiée.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
ComplexTy peColumnLength	Facultatif	65535	`ComplexTypeColumnL ength=123456;`

Certificat de l'autorité de certification approuvée

Indique au client HTTP où trouver le magasin d'approbation de vos certificats SSL. Ce paramètre est transmis au paramètre `ClientConfiguration.caFile`. Pour plus d'informations, consultez Configuration client AWS du Guide du développeur AWS SDK for C++ .

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
TrustedCerts	Facultatif	`%INSTALL_PATH%/bin`	`TrustedCerts=C:\\Program Files\\Amazon Athena ODBC Driver\\bin\\cacert.pem;`

Période de sondage minimale

Spécifie la valeur minimale en millisecondes à attendre avant d'interroger Athena sur l'état d'exécution de la requête.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
MinQueryExecutionPollingInterval	Facultatif	`100`	`MinQueryExecutionPollingInterval=200;`

Période de sondage maximale

Spécifie la valeur maximale en millisecondes à attendre avant d'interroger Athena sur l'état d'exécution de la requête.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
MaxQueryExecutionPollingInterval	Facultatif	`60000`	`MaxQueryExecutionPollingInterval=1000;`

Multiplicateur de sondages

Spécifie le facteur d'augmentation de la période de sondage. Par défaut, le sondage commence par la valeur de la période de sondage minimale et double à chaque sondage jusqu'à atteindre la valeur de la période de sondage maximale.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
QueryExecutionPollingIntervalMultiplier	Facultatif	2	`QueryExecutionPollingIntervalMultiplier=2;`

Durée maximale du sondage

Spécifie la valeur maximale en millisecondes pendant laquelle un pilote peut interroger Athena pour connaître l'état d'exécution de la requête.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
MaxPollDuration	Facultatif	`1800000`	`MaxPollDuration=1800000;`

Délai de connexion

La durée (en millisecondes) pendant laquelle la connexion HTTP attend pour établir une connexion. Cette valeur est définie pour le client Athena `ClientConfiguration.connectTimeoutMs`. Si elle n'est pas spécifiée, la valeur par défaut de curl est utilisée. Pour plus d'informations sur les paramètres de connexion, consultez Configuration client dans le Guide du développeur AWS SDK for Java .

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
ConnectionTimeout	Facultatif	`0`	`ConnectionTimeout=2000;`

Expiration de la demande

Spécifie l'expiration de la lecture du socket pour les clients HTTP. Cette valeur est définie pour le paramètre `ClientConfiguration.requestTimeoutMs` du client Athena. Pour plus

d'informations sur les paramètres, consultez Configuration client dans le Guide du développeur AWS SDK for Java .

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
RequestTimeout	Facultatif	`10000`	`RequestTimeout=30000;`

Options de proxy

Hôte proxy

Si vous demandez aux utilisateurs de passer par un proxy, utilisez ce paramètre pour définir l'hôte proxy. Ce paramètre correspond au paramètre `ClientConfiguration.proxyHost` dans le kit SDK AWS. Pour plus d'informations, consultez Configuration client AWS dans le Guide du développeur AWS SDK for C++.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
ProxyHost	Facultatif	`none`	`ProxyHost=127.0.0.1;`

Port proxy

Utilisez ce paramètre pour définir le port proxy. Ce paramètre correspond au paramètre `ClientConfiguration.proxyPort` dans le kit SDK AWS. Pour plus d'informations, consultez Configuration client AWS dans le Guide du développeur AWS SDK for C++.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
ProxyPort	Facultatif	`none`	`ProxyPort=8888;`

Nom d'utilisateur proxy

Utilisez ce paramètre pour définir le nom d'utilisateur proxy. Ce paramètre correspond au paramètre `ClientConfiguration.proxyUserName` dans le kit SDK AWS. Pour plus d'informations, consultez Configuration client AWS dans le Guide du développeur AWS SDK for C++.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
ProxyUID	Facultatif	`none`	`ProxyUID=username;`

Mot de passe proxy

Utilisez ce paramètre pour définir le mot de passe proxy. Ce paramètre correspond au paramètre `ClientConfiguration.proxyPassword` dans le kit SDK AWS. Pour plus d'informations, consultez Configuration client AWS dans le Guide du développeur AWS SDK for C++.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
ProxyPWD	Facultatif	`none`	`ProxyPWD=password;`

Hôte non proxy

Utilisez ce paramètre facultatif pour spécifier un hôte auquel le pilote se connecte sans utiliser de proxy. Ce paramètre correspond au paramètre `ClientConfiguration.nonProxyHosts` dans le kit SDK AWS. Pour plus d'informations, consultez Configuration client AWS dans le Guide du développeur AWS SDK for C++.

Le paramètre de connexion `NonProxyHost` est passé à l'option curl `CURLOPT_NOPROXY`. Pour plus d'informations sur le format `CURLOPT_NOPROXY`, voir CURLOPT_NOPROXY dans la documentation curl.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
NonProxyHost	Facultatif	none	NonProxyHost=.amazonaws.com,localhost,.example.net,.example.com;

Utiliser un proxy

Active le trafic utilisateur via le proxy spécifié.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UseProxy	Facultatif	none	UseProxy=1;

Options de journalisation

Les droits d'administrateur sont nécessaires pour modifier les paramètres décrits ici. Pour effectuer les modifications, vous pouvez utiliser la boîte de dialogue Options de journalisation de l'administrateur de sources de données ODBC ou modifier directement le registre Windows.

Niveau de journalisation

Cette option active les journaux du pilote ODBC. Dans Windows, vous pouvez utiliser le registre ou une boîte de dialogue pour activer ou désactiver la journalisation. L'option se trouve dans le chemin de registre suivant :

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Amazon Athena\ODBC\Driver
```

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
LogLevel	Facultatif	0	LogLevel=1;

Chemin d'accès au journal

Spécifie le chemin d'accès au fichier dans lequel les journaux du pilote ODBC sont stockés. Vous pouvez utiliser le registre ou une boîte de dialogue pour définir cette valeur. L'option se trouve dans le chemin de registre suivant :

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Amazon Athena\ODBC\Driver
```

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
LogPath	Facultatif	none	`LogPath=C:\Users\ username\projects \internal\trunk\;`

Utiliser AWS Logger

Spécifie si la journalisation du kit SDK AWS est activée. Spécifiez 1 pour activer, 0 pour désactiver.

Nom de chaîne de connexion	Type de paramètre	Valeur par défaut	Exemple de chaîne de connexion
UseAwsLogger	Facultatif	`0`	`UseAwsLogger=1;`

Migration vers le pilote ODBC 2.x

La plupart des paramètres de connexion ODBC 2.x Athena étant rétrocompatibles avec le pilote ODBC 1.x, vous pouvez réutiliser la majeure partie de votre chaîne de connexion existante avec le pilote ODBC 2.x Athena. Toutefois, les paramètres de connexion suivants nécessitent des modifications.

Niveau de journalisation

Bien que le pilote ODBC actuel propose une gamme d'options de journalisation disponibles, allant de `LOG_OFF (0)` à `LOG_TRACE (6)`, le pilote ODBC Amazon Athena ne possède que deux valeurs : 0 (désactivé) et 1 (activé).

Pour plus d'informations sur la journalisation du pilote ODBC 2.x, consultez Options de journalisation.

	Pilote ODBC 1.x	Pilote ODBC 2.x
Nom de chaîne de connexion	`LogLevel`	`LogLevel`
Type de paramètre	Facultatif	Facultatif
Valeur par défaut	`0`	`0`
Valeurs possibles	`0-6`	`0,1`
Exemple de chaîne de connexion	`LogLevel=6;`	`LogLevel=1;`

MetadataRetrievalMethod

Le pilote ODBC actuel propose plusieurs options pour récupérer les métadonnées d'Athena. Le pilote ODBC Amazon Athena rend la `MetadataRetrievalMethod` obsolète et utilise toujours l'API Amazon Athena pour extraire les métadonnées.

Athena introduit le drapeau `QueryExternalCatalogs` pour l'interrogation des catalogues externes. Pour interroger des catalogues externes avec le pilote ODBC actuel, définissez `MetadataRetrievalMethod` sur `ProxyAPI`. Pour interroger des catalogues externes avec le pilote ODBC, définissez `QueryExternalCatalogs` sur `1`.

	Pilote ODBC 1.x	Pilote ODBC 2.x
Nom de chaîne de connexion	`MetadataRetrievalMethod`	`QueryExternalCatalogs`
Type de paramètre	Facultatif	Facultatif
Valeur par défaut	`Auto`	`0`
Valeurs possibles	`Auto`, `AWS Glue`, `ProxyAPI`, `Query`	`0`,`1`

	Pilote ODBC 1.x	Pilote ODBC 2.x
Exemple de chaîne de connexion	`MetadataRetrievalMethod=ProxyAPI;`	`QueryExternalCatalogs=1;`

Test de connexion

Lorsque vous testez une connexion à un pilote ODBC 1.x, le pilote exécute une requête `SELECT 1` qui génère deux fichiers dans votre compartiment Amazon S3 : un pour le jeu de résultats et un pour les métadonnées. La connexion de test est facturée conformément à la politique de tarification Amazon Athena.

Lorsque vous testez une connexion à un pilote OBDC 2.x, le pilote appelle l'action d'API AthenaGetWorkGroup. L'appel utilise le type d'authentification et le fournisseur d'informations d'identification correspondant que vous avez spécifiés pour récupérer les informations d'identification. Le test de connexion est gratuit lorsque vous utilisez le pilote ODBC 2.x, et le test ne génère aucun résultat de requête dans votre compartiment Amazon S3.

Résolution des problèmes liés au pilote ODBC 2.x

Si vous rencontrez des problèmes avec le pilote ODBC Amazon Athena, vous pouvez contacter AWS Support (dans la AWS Management Console, choisissez Support, Centre de support).

Assurez-vous d'inclure les informations suivantes et de fournir tout détail supplémentaire qui aidera l'équipe d'assistance à comprendre votre cas d'utilisation.

- Description (Obligatoire) : description qui inclut des informations détaillées sur votre cas d'utilisation et la différence entre le comportement attendu et observé. Incluez toutes les informations susceptibles d'aider les ingénieurs du support à parcourir facilement le problème. Si le problème est intermittent, spécifiez les dates, les horodatages ou les points d'intervalle auxquels le problème s'est produit.
- Informations sur la version (Obligatoire) : informations sur la version du pilote, le système d'exploitation et les applications que vous avez utilisées. Par exemple, « pilote ODBC version 1.2.3, Windows 10 (x64), Power BI ».
- Fichiers journaux (Obligatoire) : nombre minimal de fichiers journaux du pilote ODBC nécessaires pour comprendre le problème. Pour plus d'informations sur les options de journalisation du pilote ODBC 2.x, consultez Options de journalisation.

- Chaîne de connexion (Obligatoire) : votre chaîne de connexion ODBC ou une capture d'écran de la boîte de dialogue qui montre les paramètres de connexion que vous avez utilisés. Pour plus d'informations sur les paramètres de connexion, consultez Paramètres de connexion ODBC 2.x Athena.
- Étapes du problème (Facultatif) : Si possible, incluez des étapes ou un programme autonome qui peut aider à reproduire le problème.
- Informations sur les erreurs de requête (Facultatif) : si vous rencontrez des erreurs impliquant des requêtes DML ou DDL, incluez les informations suivantes :
  - Une version complète ou simplifiée de la requête DML ou DDL ayant échoué.
  - L'ID du compte et la Région AWS utilisés, et l'ID d'exécution de la requête.
- Erreurs SAML (Facultatif) : si vous rencontrez un problème lié à l'authentification avec l'assertion SAML, incluez les informations suivantes :
  - Le fournisseur d'identité et le plug-in d'authentification utilisés.
  - Un exemple avec le jeton SAML.

Notes de mise à jour Amazon Athena ODBC 2.x

Ces notes de mise à jour fournissent des informations détaillées sur les améliorations, les fonctionnalités, les problèmes connus et les modifications du flux de travail du pilote Amazon Athena ODBC 2.x.

2.0.3.0

Publié le 2024-04-08

Le pilote ODBC Amazon Athena v2.0.3.0 contient les améliorations et correctifs suivants.

Améliorations

- Ajout du support MFA pour le plugin d'authentification Okta sur les plateformes Linux et Mac.
- La `athena-odbc.dll` bibliothèque et le `AmazonAthenaODBC-2.x.x.x.msi` programme d'installation pour Windows sont désormais signés.
- Le `cacert.pem` fichier de certificat CA installé avec le pilote a été mis à jour.
- Amélioration du temps nécessaire pour répertorier les tables dans les catalogues Lambda. Pour les types de `LAMBDA` catalogue, le pilote ODBC peut désormais envoyer une SHOW TABLES requête pour obtenir la liste des tables disponibles. Pour plus d'informations, consultez Utiliser la requête pour répertorier les tables.

- Le paramètre de `UseWCharForStringTypes` connexion a été introduit pour signaler les types de données de chaîne à l'aide `SQL_WCHAR` de et`SQL_WVARCHAR`. Pour plus d'informations, consultez Utiliser WCHAR pour les types de chaînes.

Correctifs

- Correction d'un avertissement de corruption du registre qui se produisait lorsque l'OdbcDsn PowerShell outil Get était utilisé.
- Mise à jour de la logique d'analyse pour gérer les commentaires au début des chaînes de requête.
- Les types de données de date et d'horodatage autorisent désormais zéro dans le champ de l'année.

Pour télécharger le nouveau pilote ODBC v2, veuillez consulter Téléchargement du pilote ODBC 2.x. Pour des informations sur la connexion, veuillez consulter Amazon Athena ODBC 2.x.

2.0.2.2

Publié le 13/02/2022

Le pilote ODBC Amazon Athena v2.0.2.2 contient les améliorations et correctifs suivants.

Améliorations

- Ajout de deux paramètres de connexion`ComplexTypeColumnLength`, `StringColumnLength` et que vous pouvez utiliser pour modifier la longueur de colonne par défaut pour les chaînes et les types de données complexes. Pour plus d'informations, consultez Longueur de colonne de chaîne et Longueur de colonne de type complexe.
- Support a été ajouté pour les systèmes d'exploitation Linux et macOS (Intel et ARM). Pour plus d'informations, consultez Linux et macOS.
- L'AWS-SDK-CPP a été mis à jour vers la version de balise 1.11.245.
- La bibliothèque curl a été mise à jour vers la version 8.6.0.

Correctifs

- Résolution d'un problème en raison duquel des valeurs incorrectes étaient signalées dans les métadonnées du jeu de résultats pour les types de données de type chaîne dans la colonne de précision.

Pour télécharger le pilote ODBC v2, veuillez consulter Téléchargement du pilote ODBC 2.x. Pour des informations sur la connexion, veuillez consulter Amazon Athena ODBC 2.x.

2.0.2.1

Date de publication : 07/12/2023

La version 2.0.2.1 du pilote ODBC Amazon Athena contient les améliorations et correctifs suivants.

Améliorations

- Amélioration de la sécurité des threads du pilote ODBC pour toutes les interfaces.
- Lorsque la journalisation est activée, les valeurs de date/heure sont désormais enregistrées avec une précision à la milliseconde.
- Lors de l'authentification avec le plug-in Navigateur SSO OIDC, le terminal s'ouvre désormais pour afficher le code de l'appareil à l'utilisateur.

Correctifs

- Résolution d'un problème de libération de mémoire qui se produisait lors de l'analyse des résultats de l'API de streaming.
- Les requêtes des interfaces `SQLTablePrivileges()`, `SQLSpecialColumns()`, `SQLProcedureColumns()` et `SQLProcedures()` renvoient désormais des ensembles de résultats vides.

Pour télécharger le pilote ODBC v2, veuillez consulter Téléchargement du pilote ODBC 2.x. Pour des informations sur la connexion, veuillez consulter Amazon Athena ODBC 2.x.

2.0.2.0

Date de publication : 17/10/2023

La version 2.0.2.0 du pilote ODBC Amazon Athena contient les améliorations et correctifs suivants.

Améliorations

- Fonctionnalité de cache de fichiers ajoutée pour le navigateur Azure AD, le navigateur SSO OIDC et les plugins d'authentification basés sur le navigateur Okta.

Les outils de BI tels que Power BI et les plugins basés sur un navigateur utilisent plusieurs fenêtres de navigateur. Le nouveau paramètre de connexion de cache de fichiers permet de mettre en cache des informations d'identification temporaires et de les réutiliser entre les multiples processus ouverts par les applications BI.

- Les applications peuvent désormais demander des informations sur le jeu de résultats après la préparation d'une instruction.
- Les délais de connexion et de demande par défaut ont été augmentés pour une utilisation avec des réseaux clients plus lents. Pour plus d'informations, consultez Délai de connexion et Expiration de la demande.
- Des remplacements de point de terminaison ont été ajoutés pour SSO et SSO OIDC. Pour plus d'informations, consultez Remplacements des points de terminaison.
- Ajout d'un paramètre de connexion pour transmettre un argument URI pour une demande d'authentification à Ping. Vous pouvez utiliser ce paramètre pour contourner la limitation du rôle unique de Lake Formation. Pour plus d'informations, consultez Paramètre URI Ping.

Correctifs

- Correction d'un problème de dépassement d'entiers qui se produisait lors de l'utilisation du mécanisme de liaison basé sur les lignes.
- Le délai d'expiration a été supprimé de la liste des paramètres de connexion requis pour le plug-in d'authentification du navigateur SSO OIDC.
- Ajout d'interfaces manquantes pour `SQLStatistics()`, `SQLPrimaryKeys()`, `SQLForeignKeys()` et `SQLColumnPrivileges()`, et ajout de la possibilité de renvoyer des ensembles de résultats vides sur demande.

Pour télécharger le nouveau pilote ODBC v2, veuillez consulter Téléchargement du pilote ODBC 2.x. Pour des informations sur la connexion, veuillez consulter Amazon Athena ODBC 2.x.

2.0.1.1

Date de publication : 10/08/2023

La version 2.0.1.1 du pilote ODBC Amazon Athena contient les améliorations et correctifs suivants.

Améliorations

- Ajout de la journalisation URI au plug-in d'authentification Okta.
- Le paramètre de rôle préféré a été ajouté au plug-in du fournisseur d'informations d'identification externe.
- Ajout de la gestion du préfixe de profil dans le nom de profil du fichier de configuration AWS .

Correctifs

- Correction d'un problème d' Région AWS utilisation survenu lors de la collaboration avec Lake Formation et AWS STS ses clients.
- Restauration des clés de partition manquantes dans la liste des colonnes de la table.
- Ajout du type d'authentification `BrowserSSOOIDC` manquant au profil AWS .

Pour télécharger le nouveau pilote ODBC v2, consultez Téléchargement du pilote ODBC 2.x.

2.0.1.0

Date de publication : 29/06/2023

Amazon Athena publie la version 2.0.1.0 du pilote ODBC.

Athena a publié un nouveau pilote ODBC qui améliore l'expérience de connexion, d'interrogation et de visualisation des données à partir d'applications de développement SQL et de business intelligence compatibles. La dernière version du pilote ODBC Athena prend en charge les fonctionnalités du pilote existant et est facile à mettre à niveau. La nouvelle version inclut la prise en charge de l'authentification des utilisateurs via AWS IAM Identity Center. Elle offre également la possibilité de lire les résultats des requêtes à partir d'Amazon S3, ce qui permet de les mettre à votre disposition plus rapidement.

Pour plus d'informations, voir Amazon Athena ODBC 2.x.

Pilote ODBC 1.x d'Athena

Utilisez les liens sur cette page pour télécharger le contrat de licence du pilote ODBC 1.x d'Amazon Athena, les pilotes ODBC et la documentation ODBC. Pour des informations relatives à la chaîne de connexion ODBC, consultez le fichier PDF du Guide de configuration et d’installation du pilote ODBC,

téléchargeable à partir de cette page. Pour plus d'informations sur les autorisations, voir Accès via les connexions JDBC et ODBC.

> **Important**
>
> Lorsque vous utilisez le pilote ODBC 1.x, veillez à respecter les exigences suivantes :
>
> - Ouvrez le port 444 – Conservez ouvert le port 444, utilisé par Athena pour diffuser les résultats de requête, pour le trafic sortant. Lorsque vous utilisez un PrivateLink point de terminaison pour vous connecter à Athena, assurez-vous que le groupe de sécurité attaché au PrivateLink point de terminaison est ouvert au trafic entrant sur le port 444.
> - athena : GetQueryResultsStream policy — Ajoutez l'action `athena:GetQueryResultsStream` de stratégie aux principaux IAM qui utilisent le pilote ODBC. Cette action de politique n'est pas exposée directement avec l'API. Elle est uniquement utilisée avec les pilotes ODBC et JDBC dans le cadre de la prise en charge des résultats de streaming. Pour un exemple de politique, consultez AWS politique gérée : AWSQuicksightAthenaAccess.

Windows

Versions du pilote	Lien de téléchargement
ODBC 1.2.3.1000 pour Windows 32 bits	Pilote ODBC Windows 32 bits 1.2.3.1000 Pilote ODBC bits 1.2.3.1000
ODBC 1.2.3.1000 pour Windows 64 bits	Pilote ODBC pour Windows 64 bits 1.2.3.1000 Pilote ODBC pour bits 1.2.3.1000

Linux

Versions du pilote	Lien de téléchargement
ODBC 1.2.3.1000 pour Linux 32 bits	Pilote ODBC Linux 32 bits 1.2.3.1000 Pilote ODBC bits 1.2.3.1000

Versions du pilote	Lien de téléchargement
ODBC 1.2.3.1000 pour Linux 64 bits	Pilote ODBC Linux 64 bits 1.2.3.1000 Pilote ODBC bits 1.2.3.1000

OSX

Versions du pilote	Lien de téléchargement
ODBC 1.2.3.1000 pour OSX	

Documentation

Contenu	Lien vers la documentation
Contrat de licence du pilote ODBC Amazon Athena	Contrat de licence
Documentation pour ODBC 1.2.3.1000	Guide d'installation et de configuration du pilote ODBC version 1.2.3.1000 Guide d' version 1.2.3.1000
Notes de mise à jour pour ODBC 1.2.3.1000	Notes de mise à jour du pilote ODBC version 1.2.3.1000 Notes de mise à jour du pilote version 1.2.3.1000

Notes du pilote ODBC

Connexion sans utiliser de proxy

Si vous souhaitez spécifier certains hôtes auxquels le pilote se connecte sans utiliser de proxy, vous pouvez utiliser la propriété `NonProxyHost` facultative dans votre chaîne de connexion ODBC.

La propriété `NonProxyHost` spécifie une liste d'hôtes, séparés par des virgules, auxquels le connecteur peut accéder sans passer par le serveur proxy lorsqu'une connexion proxy est activée, comme dans l'exemple suivant :

```
.amazonaws.com,localhost,.example.net,.example.com
```

Le paramètre de connexion `NonProxyHost` est passé à l'option curl `CURLOPT_NOPROXY`. Pour plus d'informations sur le format `CURLOPT_NOPROXY`, voir CURLOPT_NOPROXY dans la documentation curl.

Configuration de l'accès fédéré à Amazon Athena pour les utilisateurs de Microsoft AD FS à l'aide d'un client ODBC

Pour configurer l'accès fédéré à Amazon Athena pour les utilisateurs de Microsoft Active Directory Federation Services (AD FS) à l'aide d'un client ODBC, vous devez d'abord établir la confiance entre AD FS et votre compte AWS. Une fois cette confiance établie, vos utilisateurs AD peuvent se fédérer enAWS utilisant leurs informations d'identification AD et obtenir les autorisations d'un rôle AWS Identity and Access Management (IAM) pour accéder à des ressources AWS telles que l'API Athena.

Pour créer cette confiance, vous ajoutez AD FS en tant que fournisseur SAML à votre Compte AWS et créez un rôle IAM que les utilisateurs fédérés peuvent assumer. Du côté d'AD FS, vous ajoutez en AWS tant que partie utilisatrice et vous rédigez des règles de revendication SAML pour envoyer les bons attributs d'utilisateur à AWS pour autorisation (en particulier, Athena et Amazon S3).

La configuration de l'accès AD FS à Athena implique les principales étapes suivantes :

1. Configuration d'un fournisseur et d'un rôle IAM SAML

2. Configuration d'AD FS

3. Création d'utilisateurs et de groupes Active Directory

4. Configuration de la connexion ODBC AD FS à Athena

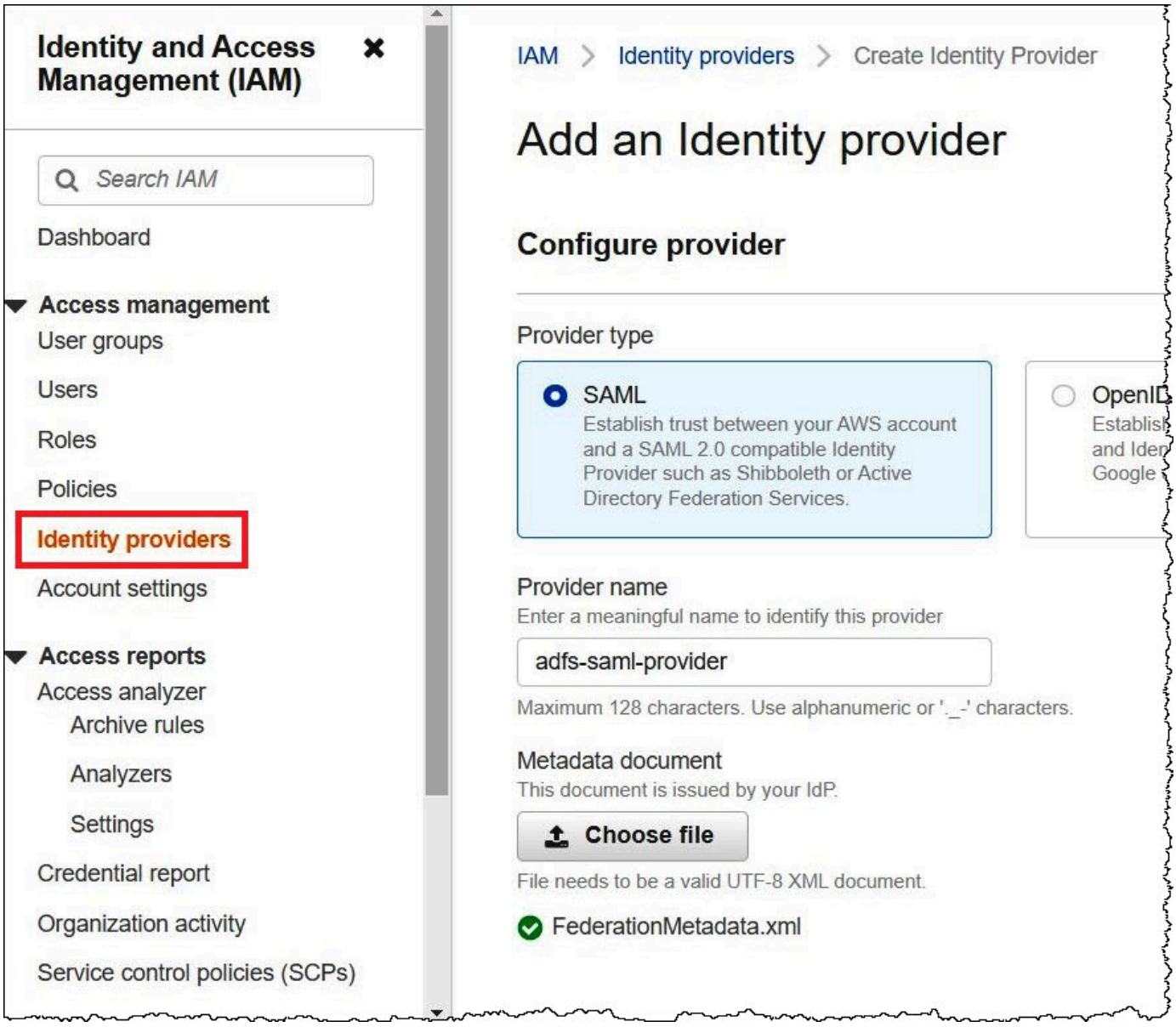
1. Configuration d'un fournisseur et d'un rôle IAM SAML

Dans cette section, vous ajoutez AD FS en tant que fournisseur SAML à votre compte AWS et créez un rôle IAM que vos utilisateurs fédérés peuvent assumer.

Configurer un fournisseur SAML

1. Connectez-vous à l'outil AWS Management Console, puis ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation, sélectionnez Fournisseurs d'identité.
3. Choisissez Ajouter un fournisseur.

4. Pour Type de fournisseur, choisissez SAML.



5. Pour Provider name (Nom du fournisseur), saisissez **`adfs-saml-provider`**.

6. Dans un navigateur, saisissez l'adresse suivante pour télécharger le fichier XML de fédération pour votre serveur AD FS. Pour effectuer cette étape, votre navigateur doit avoir accès au serveur AD FS.

```
https://adfs-server-name/federationmetadata/2007-06/federationmetadata.xml
```

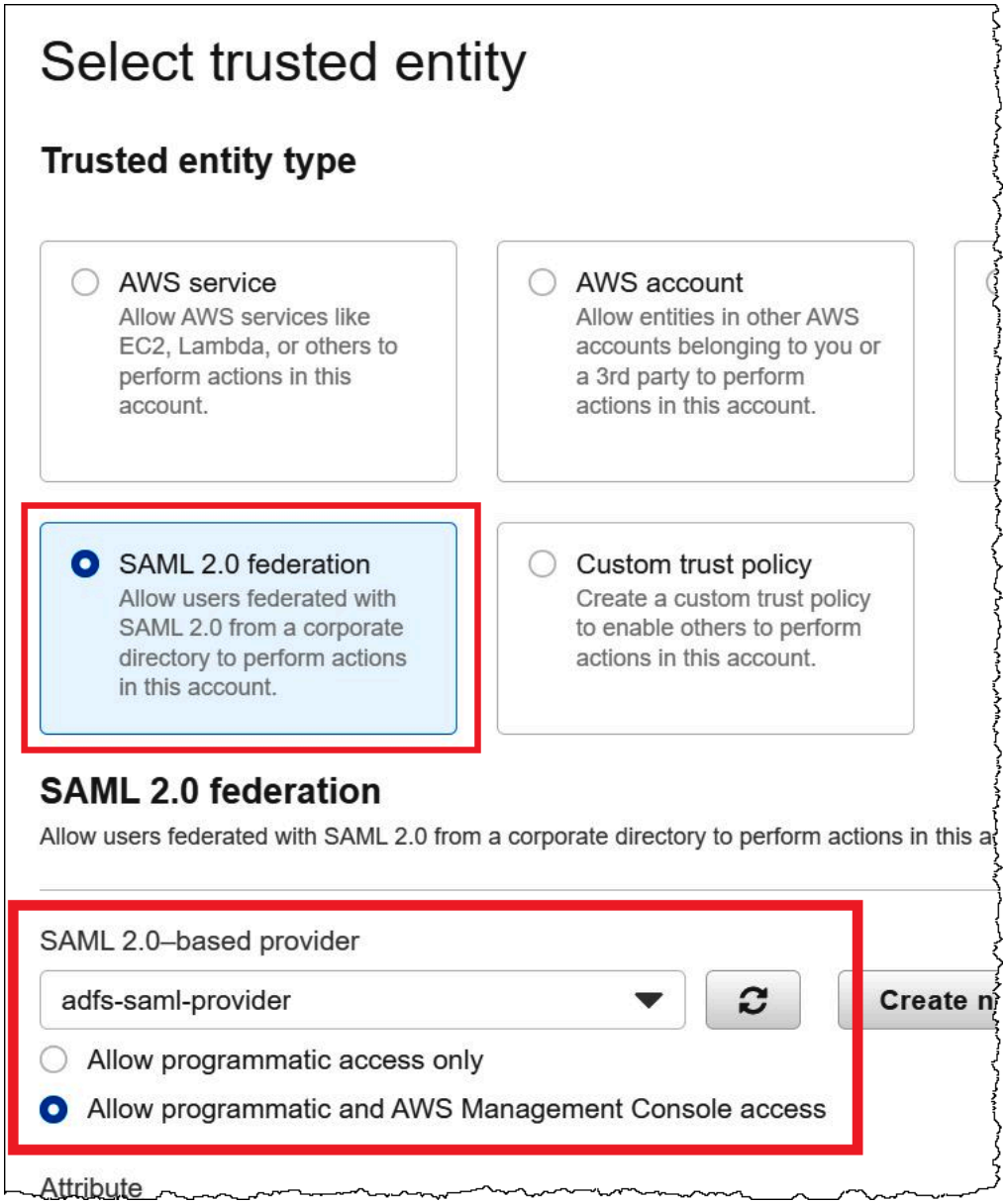
7. Dans la console IAM, pour Metadata document (Document de métadonnées), sélectionnez Choose file (Choisir un fichier), puis téléchargez le fichier de métadonnées de fédération vers AWS.

8. Pour terminer, sélectionnez Add provider (Ajouter un fournisseur).

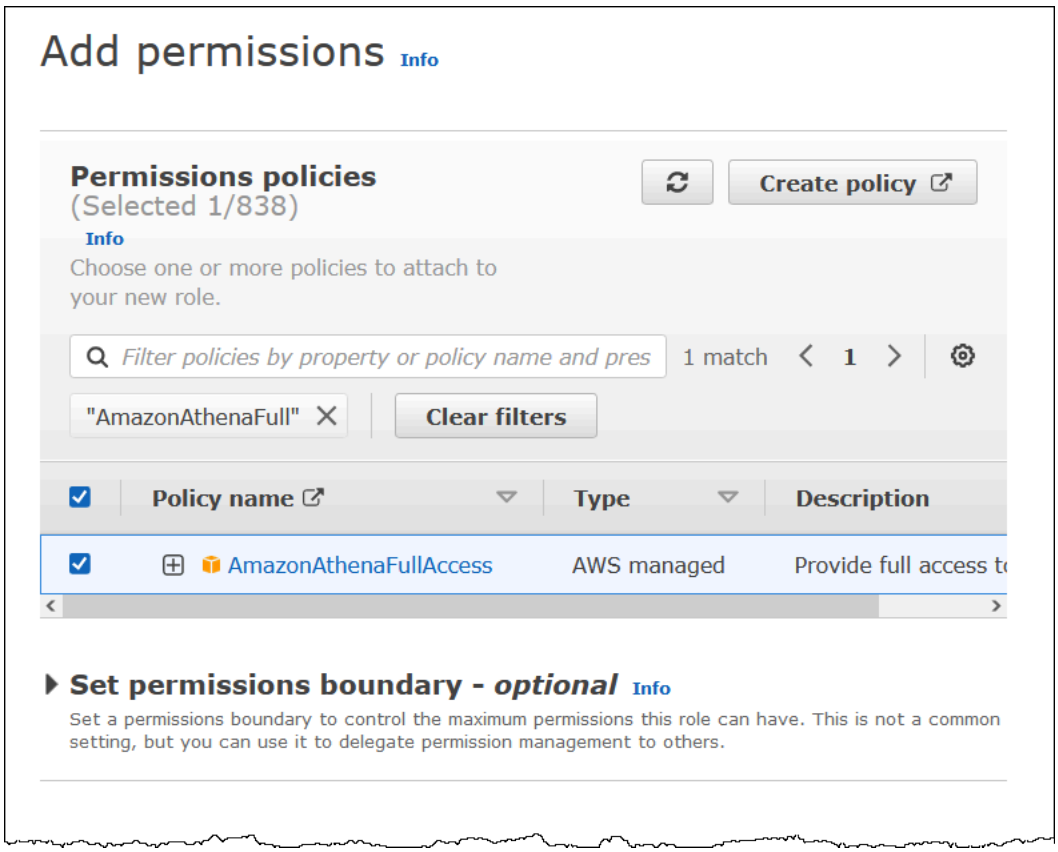
Vous créez ensuite le rôle IAM que vos utilisateurs fédérés peuvent assumer.

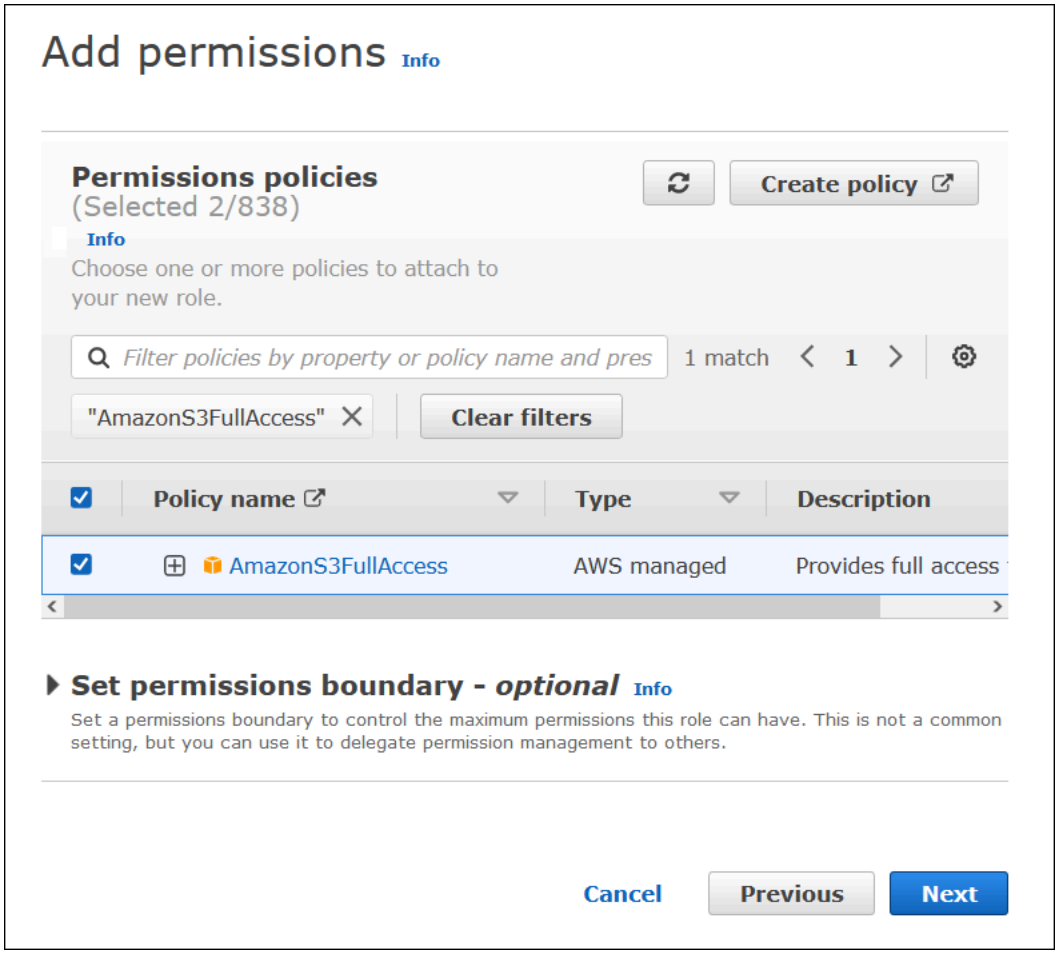
Créer un rôle IAM pour les utilisateurs fédérés

1. Dans le volet de navigation de la console IAM, sélectionnez Roles (Rôles).
2. Sélectionnez Créer un rôle.
3. Pour Trusted entity type (Type d'entité de confiance), sélectionnez SAML 2.0 Federation (Fédération SAML 2.0).
4. Pour SAML 2.0-based provider (Fournisseur basé sur SAML 2.0), choisissez le fournisseur adfs-saml-provider que vous avez créé.
5. Sélectionnez Autoriser l'accès programmatique et l'accès à la console de gestion AWS, puis sélectionnez Suivant.



6. Sur la page Add permissions (Ajouter des autorisations), filtrez les politiques d'autorisations IAM dont vous avez besoin pour ce rôle, puis cochez les cases correspondantes. Ce tutoriel attache les politiques `AmazonAthenaFullAccess` et `AmazonS3FullAccess`.


Add permissions Info
Permissions policies
(Selected 1/838)
Info
Choose one or more policies to attach to your new role.
Create policy
Filter policies by property or policy name and pres
1 match
1
"AmazonAthenaFull"
Clear filters
Policy name
Type
Description
AmazonAthenaFullAccess
AWS managed
Provide full access t
Set permissions boundary - optional Info
Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting, but you can use it to delegate permission management to others.

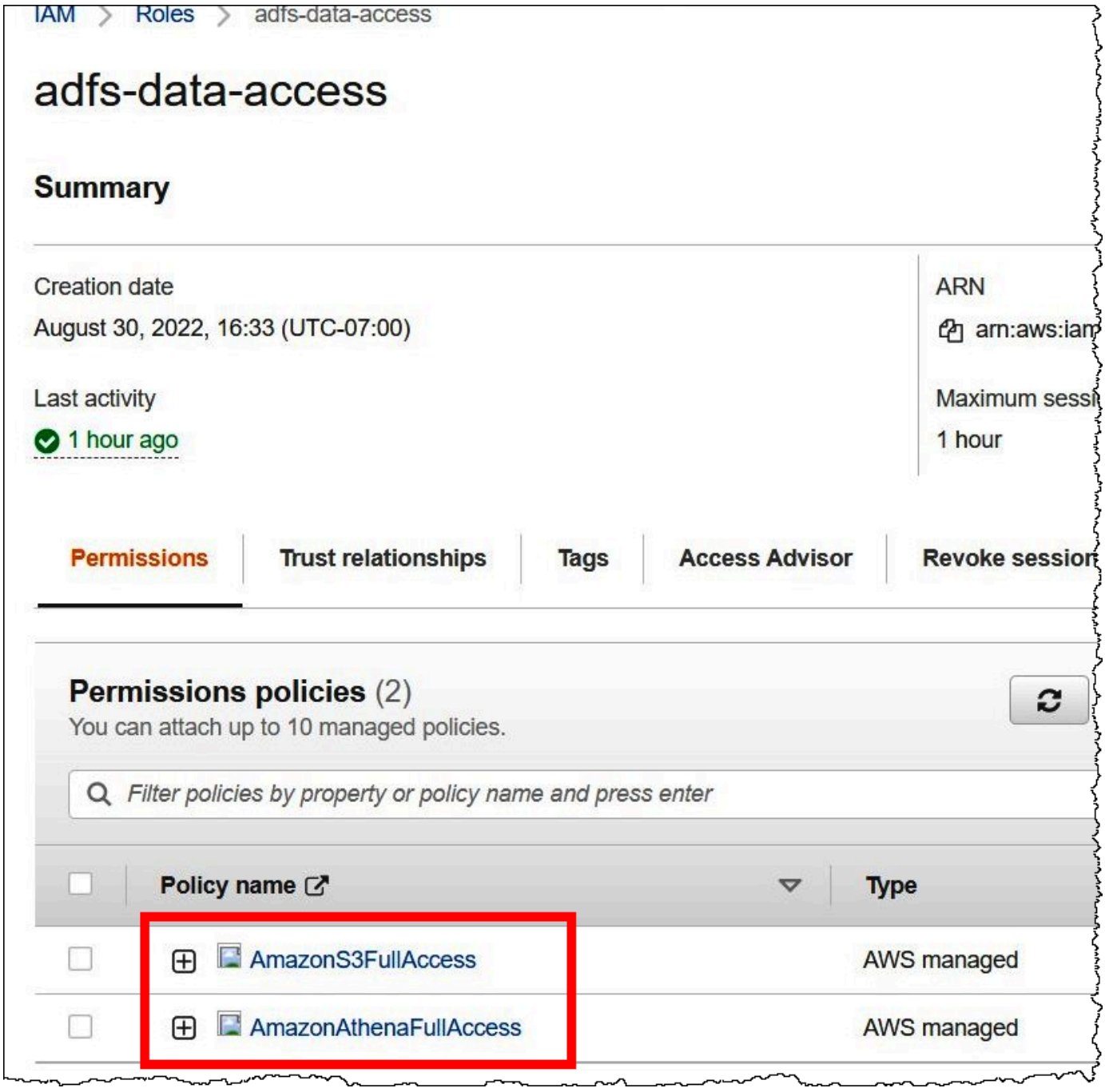


7. Choisissez Suivant.
8. Sur la page Name, review, and create (Nommer, vérifier et créer), pour Role name Role name, saisissez un nom pour le rôle. Ce tutoriel utilise le nom adfs-data-access.

   À l'étape 1 : sélectionner les entités de confiance, le champ Principal doit être automatiquement renseigné avec `"Federated:" "arn:aws:iam::account_id:saml-provider/adfs-saml-provider"`. Le champ `Condition` doit contenir `"SAML:aud"` et `"https://signin.aws.amazon.com/saml"`.

L'étape 2 : ajouter des autorisations affiche les politiques que vous avez attachées au rôle.

9. Sélectionnez Créer un rôle. Un message de type bannière confirme la création du rôle.
10. Sur la page Roles (Rôles), sélectionnez le rôle que vous venez de créer. La page récapitulative du rôle montre les politiques qui ont été attachées.



2. Configuration d'AD FS

Vous pouvez maintenant ajouter AWS en tant que partie utilisatrice et écrire des règles de revendication SAML afin d'envoyer les bons attributs d'utilisateur à AWS pour autorisation.

La fédération basée sur SAML comprend deux parties participantes : l'IdP (Active Directory) et la partie utilisatrice (AWS), qui est le service ou l'application qui utilise l'authentification à partir de l'IdP.

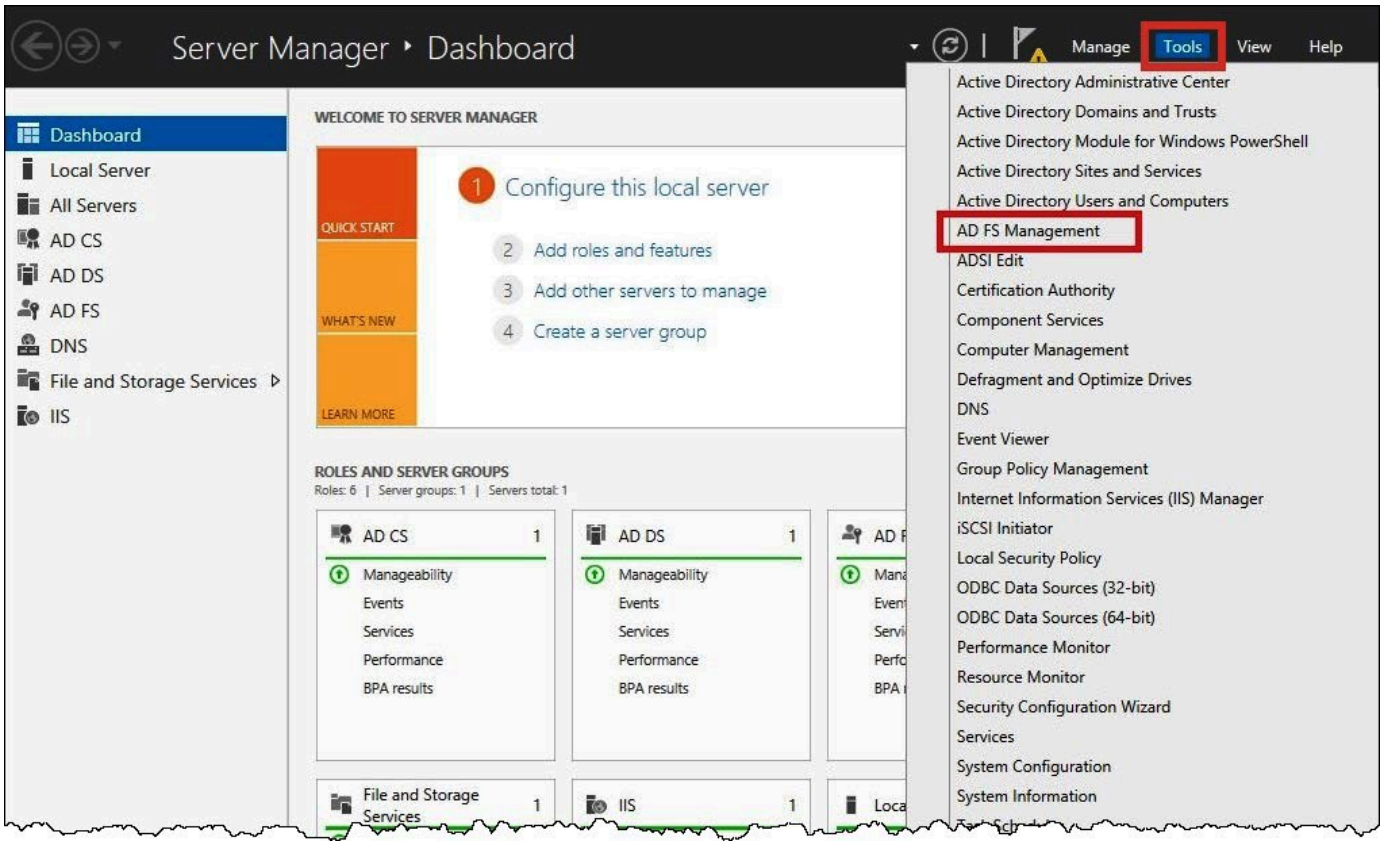
Pour configurer AD FS, vous devez d'abord ajouter une partie utilisatrice, puis configurer les règles de revendication SAML pour cette partie utilisatrice. AD FS utilise des règles de revendication pour former une assertion SAML qui est envoyée à une partie utilisatrice. L'assertion SAML indique que les informations concernant l'utilisateur AD sont vraies et que l'utilisateur a été authentifié.

Ajout d'une relation de confiance d'une partie utilisatrice

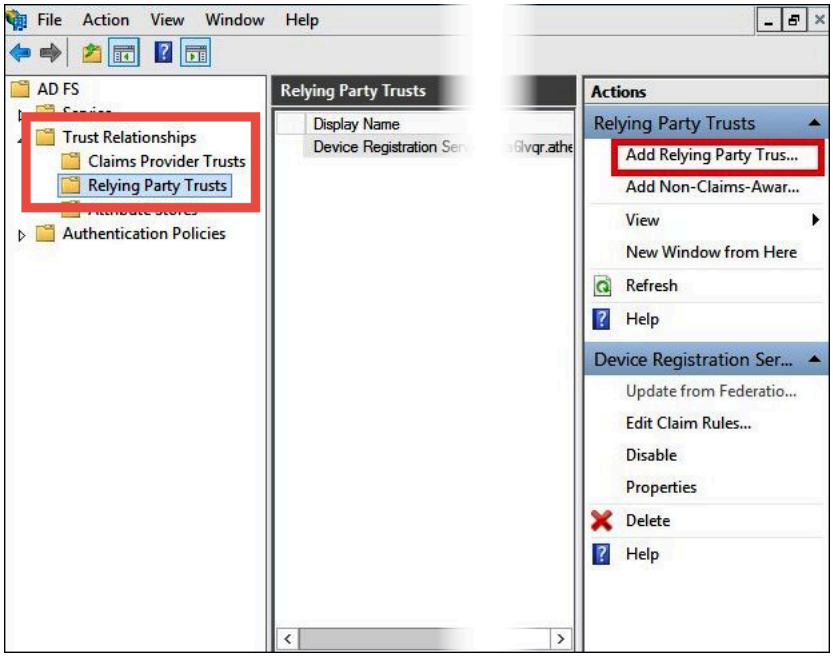
Pour ajouter une relation de confiance d'une partie utilisatrice à AD FS, vous devez utiliser le gestionnaire de serveur AD FS.

Ajouter une relation de confiance d'une partie utilisatrice dans AD FS

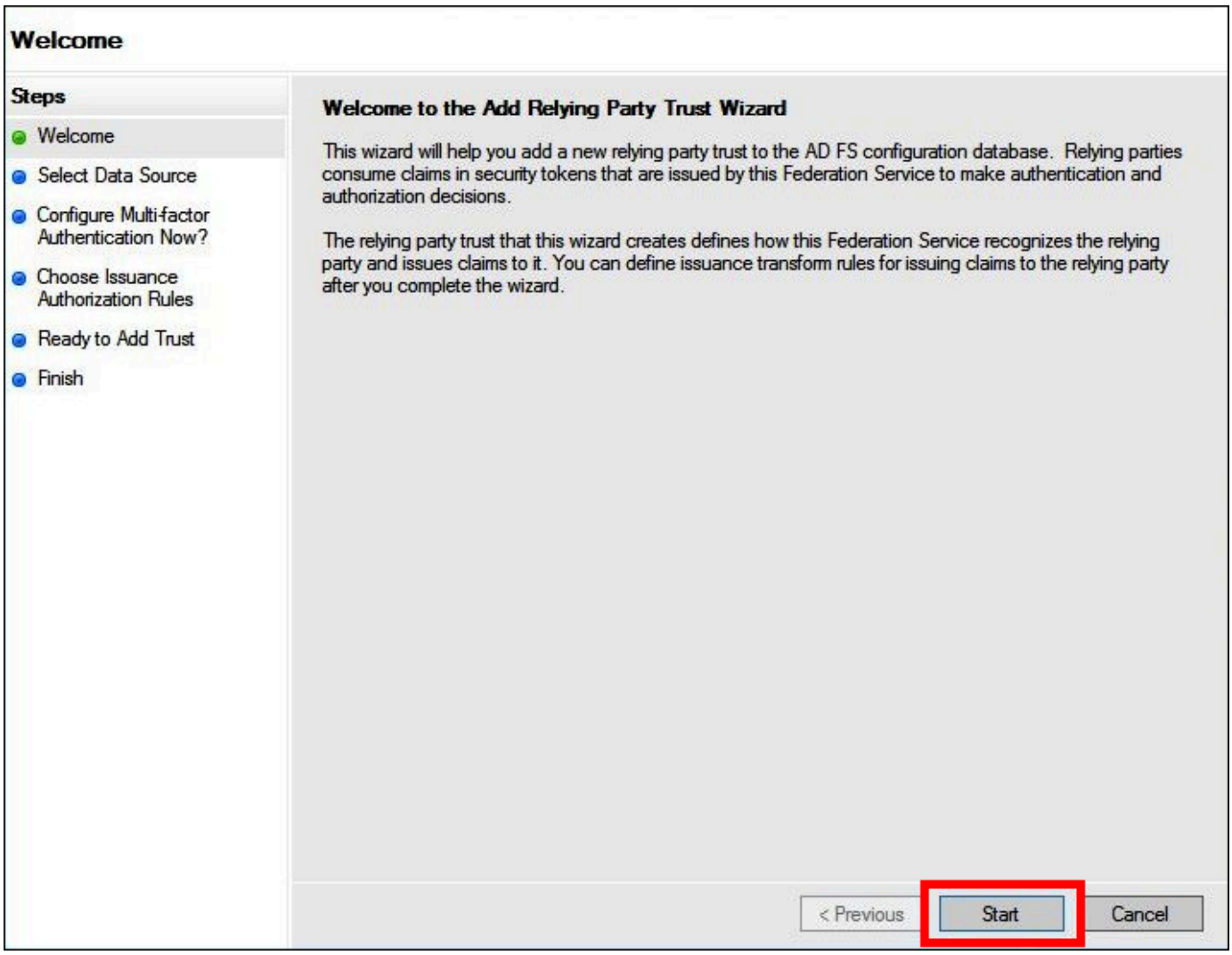
1. Connectez-vous au serveur AD FS.
2. Dans le menu Start (Démarrer), ouvrez Server Manager (Gestionnaire de serveur).
3. Choisissez Tools (Outils), puis AD FS Management (Gestion d'AD FS).



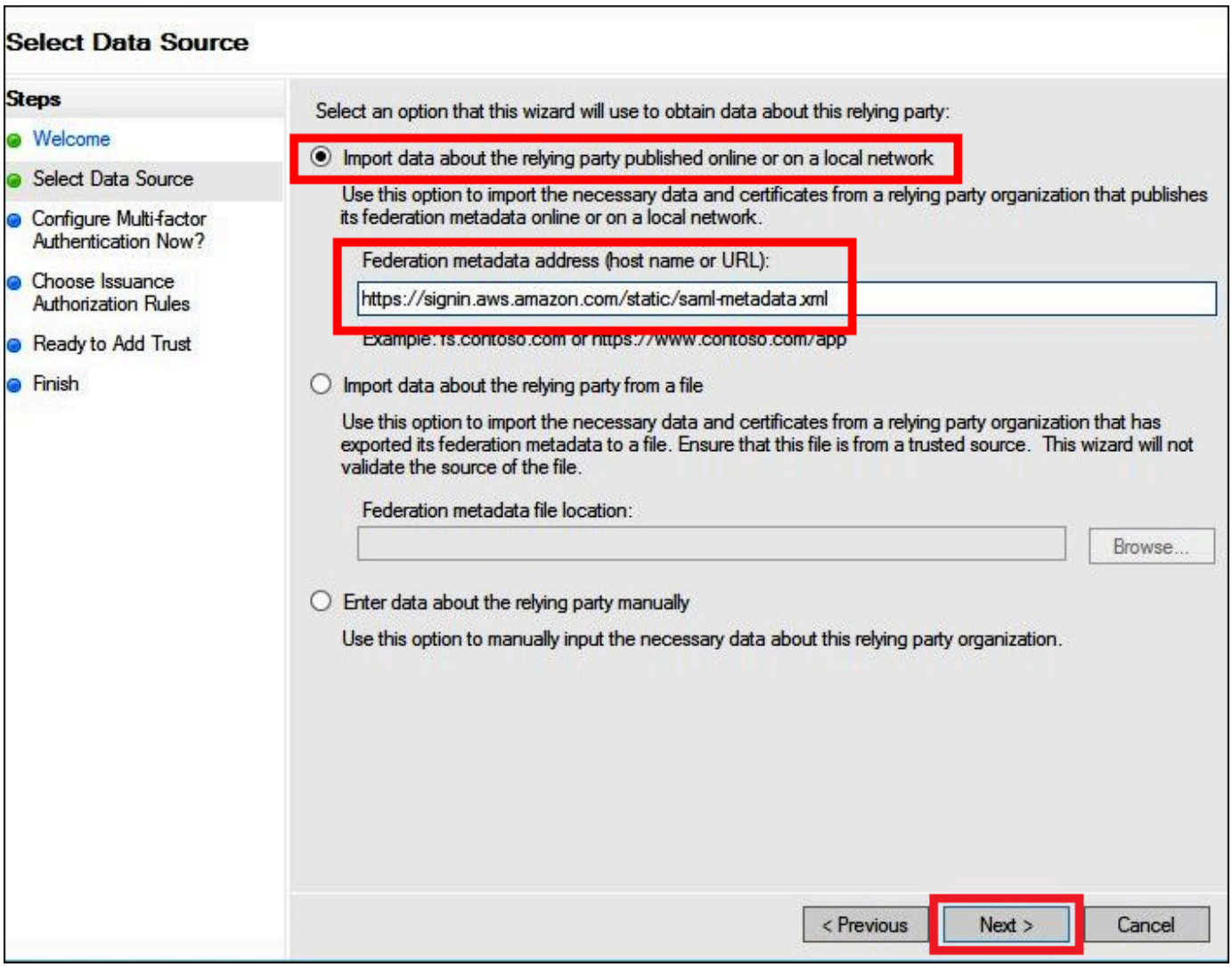
4. Dans le volet de navigation, sous Trust Relationships (Relations de confiance), choisissez Relying Party Trusts (Relations de confiance de partie utilisatrice).
5. Sous Actions, choisissez Add Relying Party Trust (Ajouter une relation de confiance d'une partie utilisatrice).



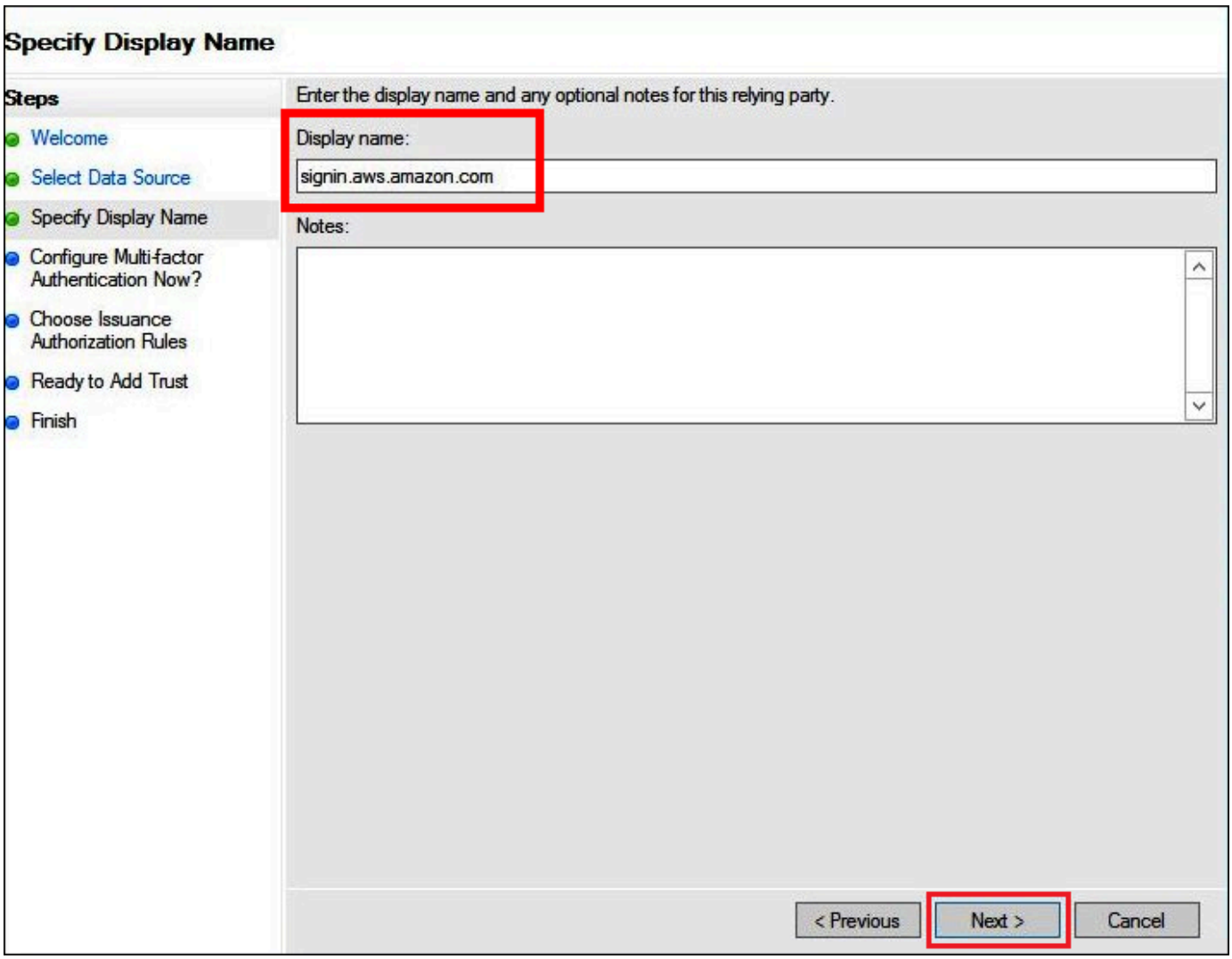
6. Dans la page Assistant Ajout d'approbation de partie de confiance, choisissez Démarrer.



7. Sur l'écran Select Data Source (Sélectionner la source de données), sélectionnez Import data about the relying party published online or on a local network (Importer les données relatives à la partie utilisatrice publiées en ligne ou sur un réseau local).
8. Pour Federation metadata address (host name or URL) (Adresse de métadonnées de fédération [nom d'hôte ou URL]), saisissez l'URL **`https://signin.aws.amazon.com/static/saml-metadata.xml`**.
9. Choisissez Next (Suivant).

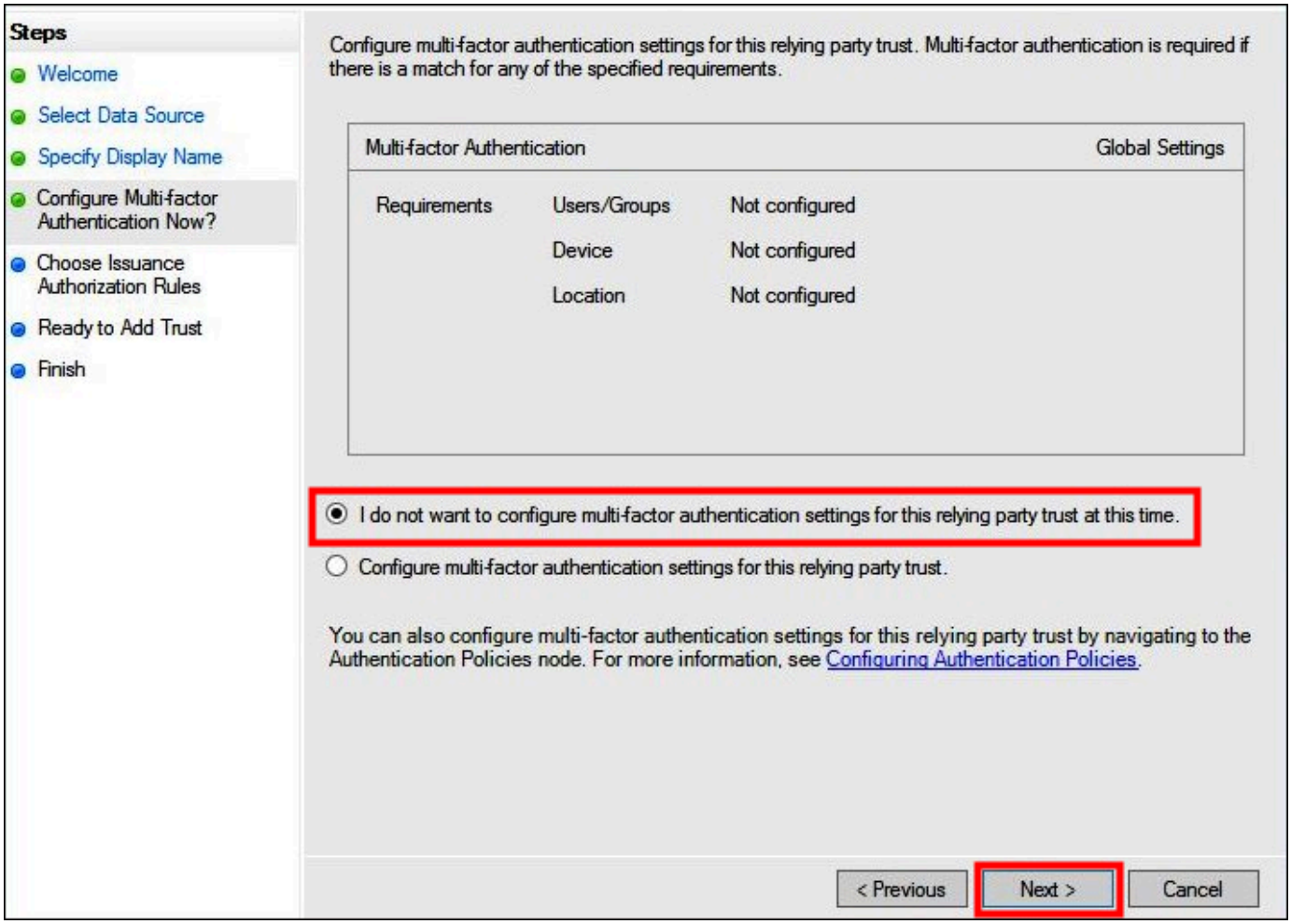


10. Sur la page Specify Display Name (Spécifier le nom d'affichage), dans le champ Display name (Nom d'affichage), saisissez un nom d'affichage pour votre partie utilisatrice, puis cliquez sur Next (Suivant).



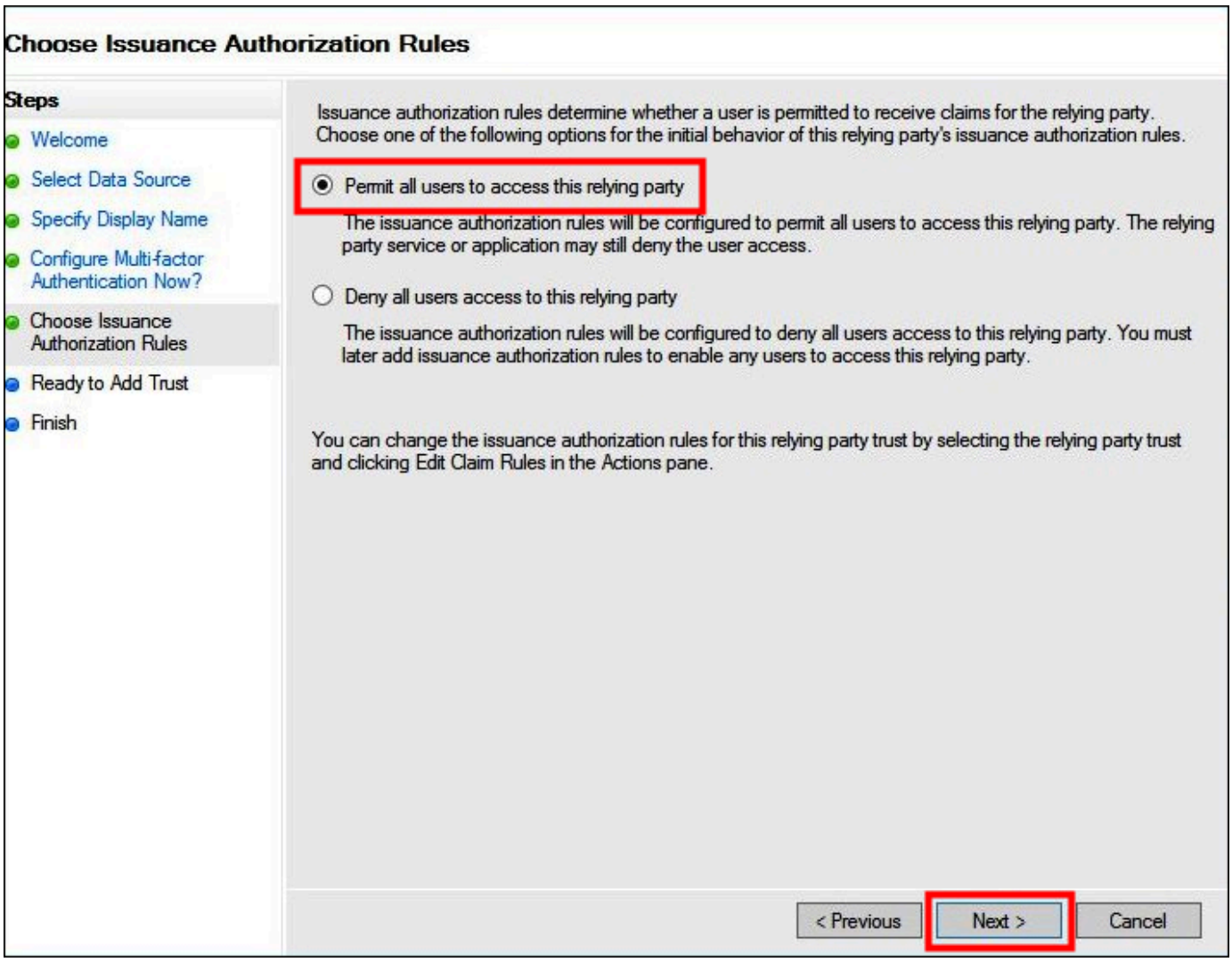
11. Sur la page Configure Multi-factor Authentication Now (Configurer l'authentification multifactorielle maintenant), ce tutoriel sélectionne I do not want to configure multi-factor authentication for this relying party trust at this time (Je ne veux pas configurer l'authentification multifactorielle pour cette partie utilisatrice en ce moment).

    Pour une sécurité accrue, nous vous recommandons de configurer une authentification multifactorielle afin de protéger vos ressources AWS. Parce qu'il utilise un exemple de jeu de données, ce tutoriel ne permet pas l'authentification multifactorielle.

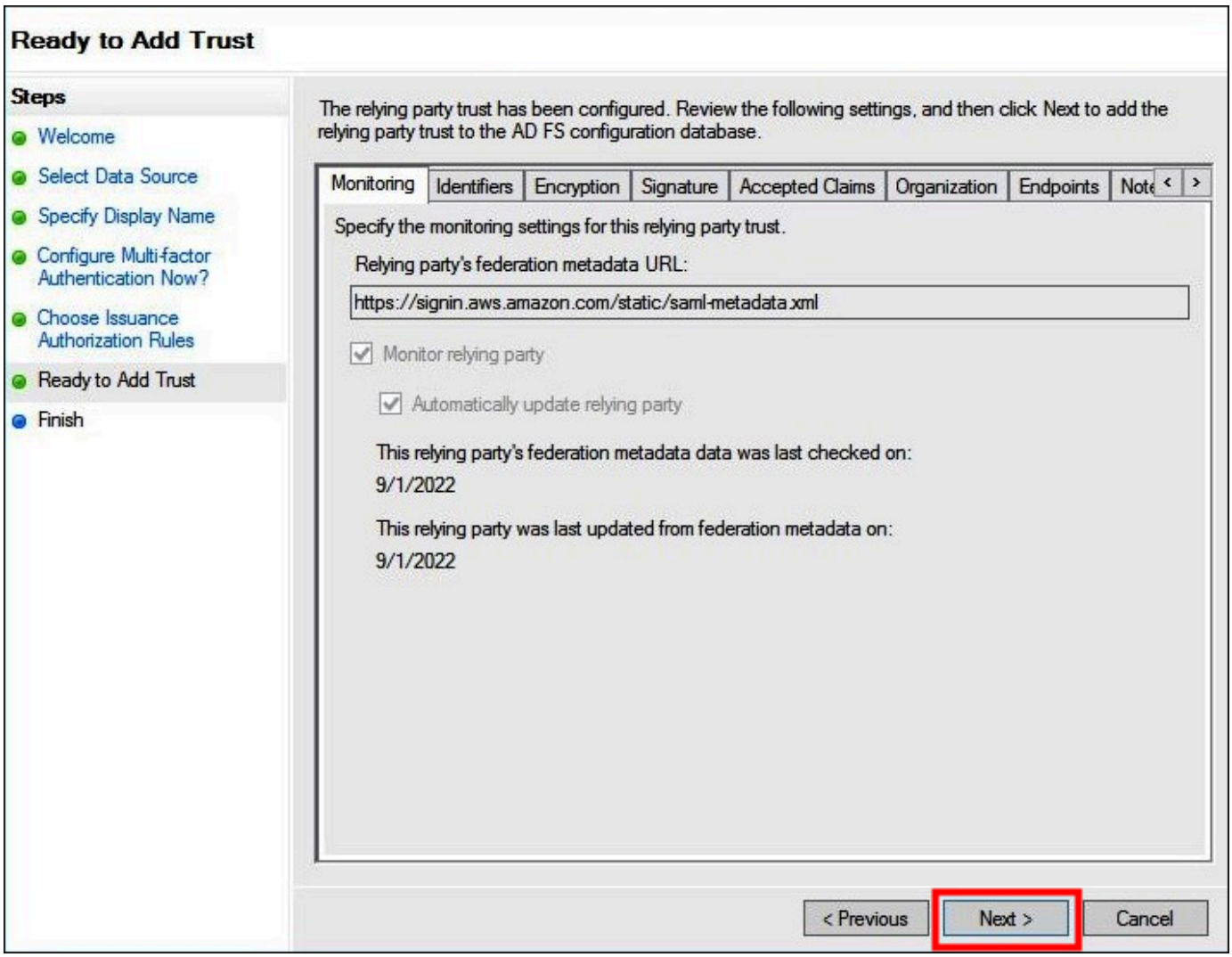


12. Choisissez Suivant.
13. Sur la page Choose Issuance Authorization Rules (Choisir les règles d'autorisation d'émission), sélectionnez Permit all users to access this relying party (Autoriser tous les utilisateurs à accéder à cette partie utilisatrice).

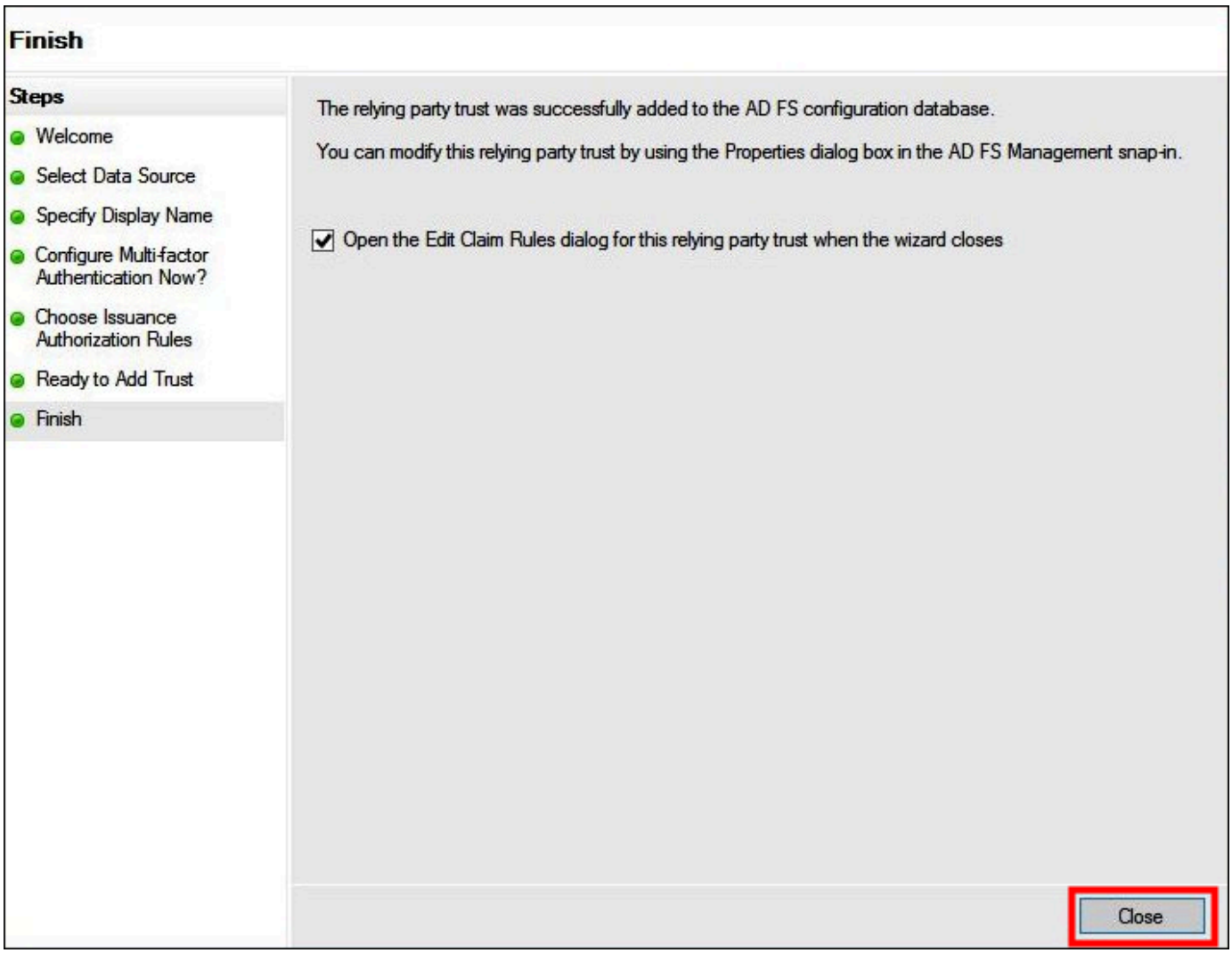
    Cette option permet à tous les utilisateurs d'Active Directory d'utiliser AD FS avec AWS en tant que partie utilisatrice. Vous devez tenir compte de vos besoins en matière de sécurité et adapter cette configuration en conséquence.



14. Choisissez Suivant.
15. Sur la page Ready to Add Trust (Prêt à ajouter une relation de confiance), cliquez sur Next (Suivant) pour ajouter la relation de confiance de la partie utilisatrice à la base de données de la configuration AD FS.



16. Sur la page Finish (Terminer), choisissez Close (Fermer).



Configuration des règles de revendication SAML pour la partie utilisatrice

Dans cette tâche, vous allez créer deux ensembles de règles de revendication.

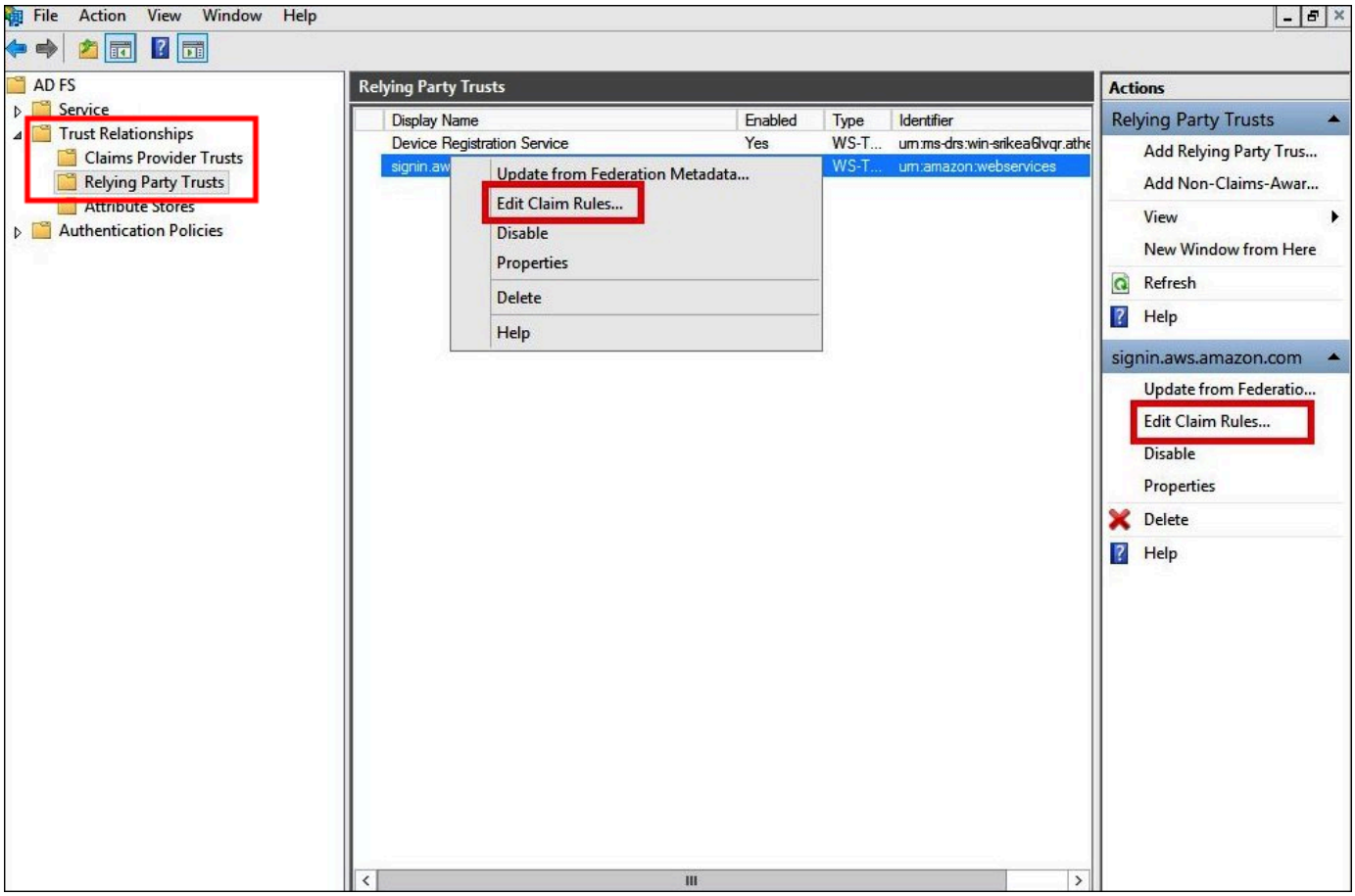
Le premier ensemble, les règles 1 à 4, contient les règles de revendication AD FS qui sont requises pour assumer un rôle IAM en fonction de l'appartenance à un groupe AD. Il s'agit des mêmes règles que vous créez si vous souhaitez établir un accès fédéré à la AWS Management Console.

Le second ensemble, les règles 5 à 6, concerne les règles de revendication requises pour le contrôle d'accès Athena.

Créer des règles de revendication AD FS

1. Dans le volet de navigation de la console de gestion AD FS, choisissez Trust Relationships (Relations de confiance), Relying Party Trusts (Relations de confiance de partie utilisatrice).

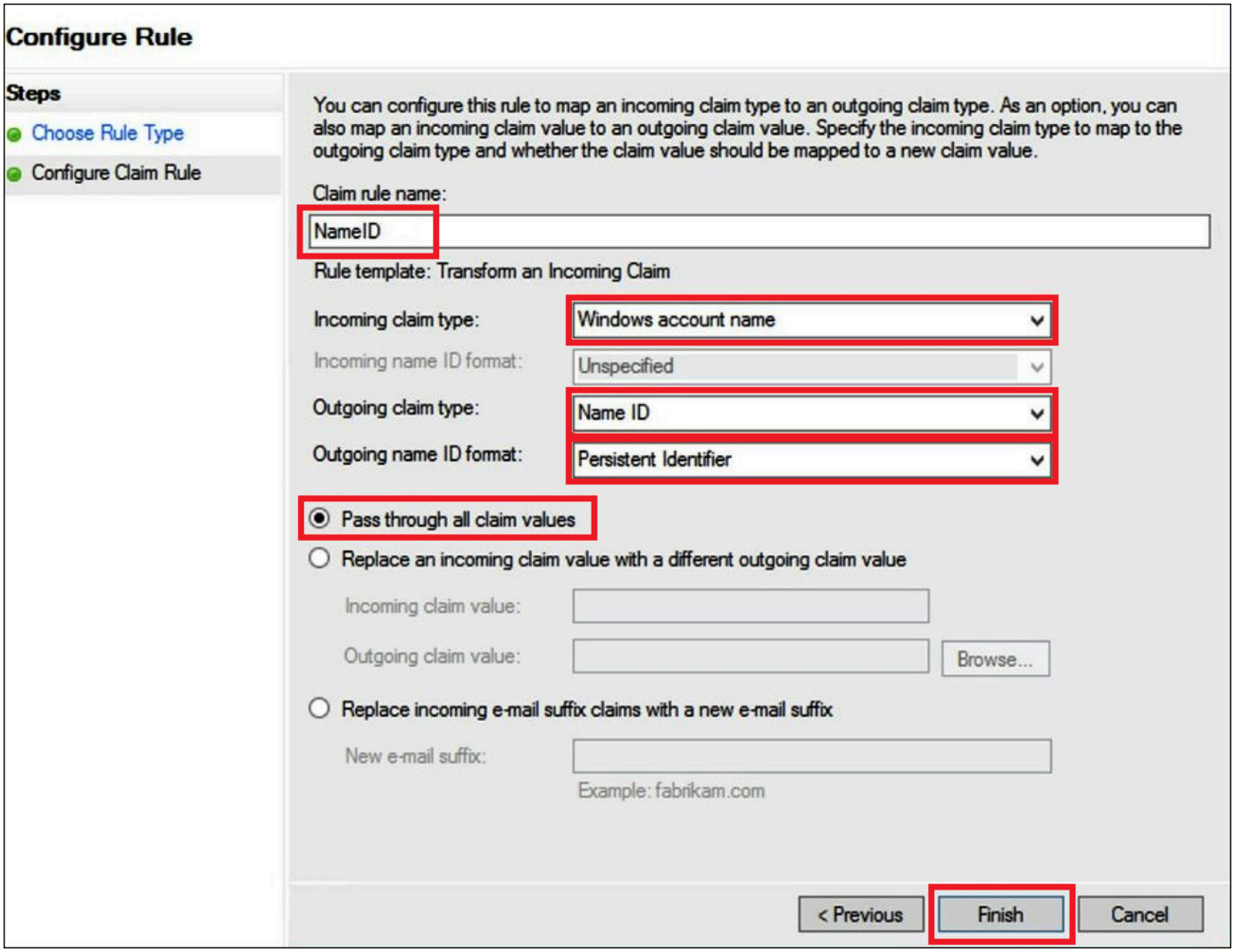
2. Trouvez la partie utilisatrice que vous avez créée dans la section précédente.
3. Faites un clic droit sur la partie utilisatrice et choisissez Edit Claim Rules (Modifier les règles de revendication), ou choisissez Edit Claim Rules (Modifier les règles de revendication) dans le menu Actions.



4. Choisissez Add Rule (Ajouter une règle).
5. Sur la page Configure Rule (Configurer une règle) de l'assistant d'ajout de règle de transformation, saisissez les informations suivantes pour créer la règle de revendication 1, puis choisissez Finish (Terminer).

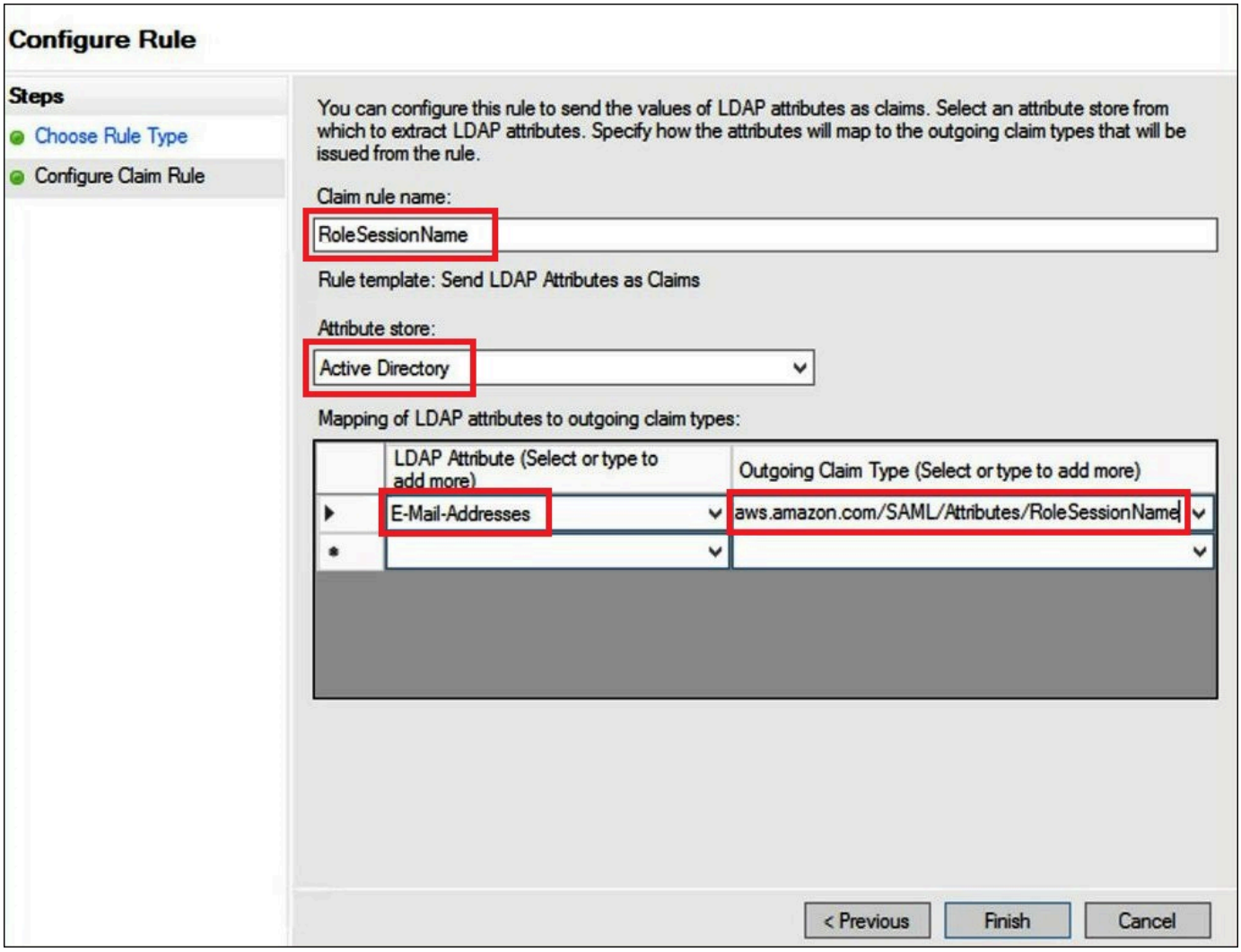
   - Pour Claim Rule name (Nom de la règle de revendication), saisissez **`NameID`**.
   - Pour Rule template (Modèle de règle), utilisez Transform an Incoming Claim (Transformer une revendication entrante).
   - Pour Incoming claim type (Nom de la revendication entrante), choisissez Windows account name (Nom du compte Windows).
   - Pour Outgoing claim type (Nom de la revendication sortante), choisissez Name ID (ID nom).
   - Pour Format d'ID de nom sortant, choisissez Identifiant persistant.

- Choisissez Pass through all claim values (Transférer toutes les valeurs de revendication).



6. Choisissez Add Rule (Ajouter une règle), saisissez les informations suivantes pour créer la règle de revendication 2, puis choisissez Finish (Terminer).

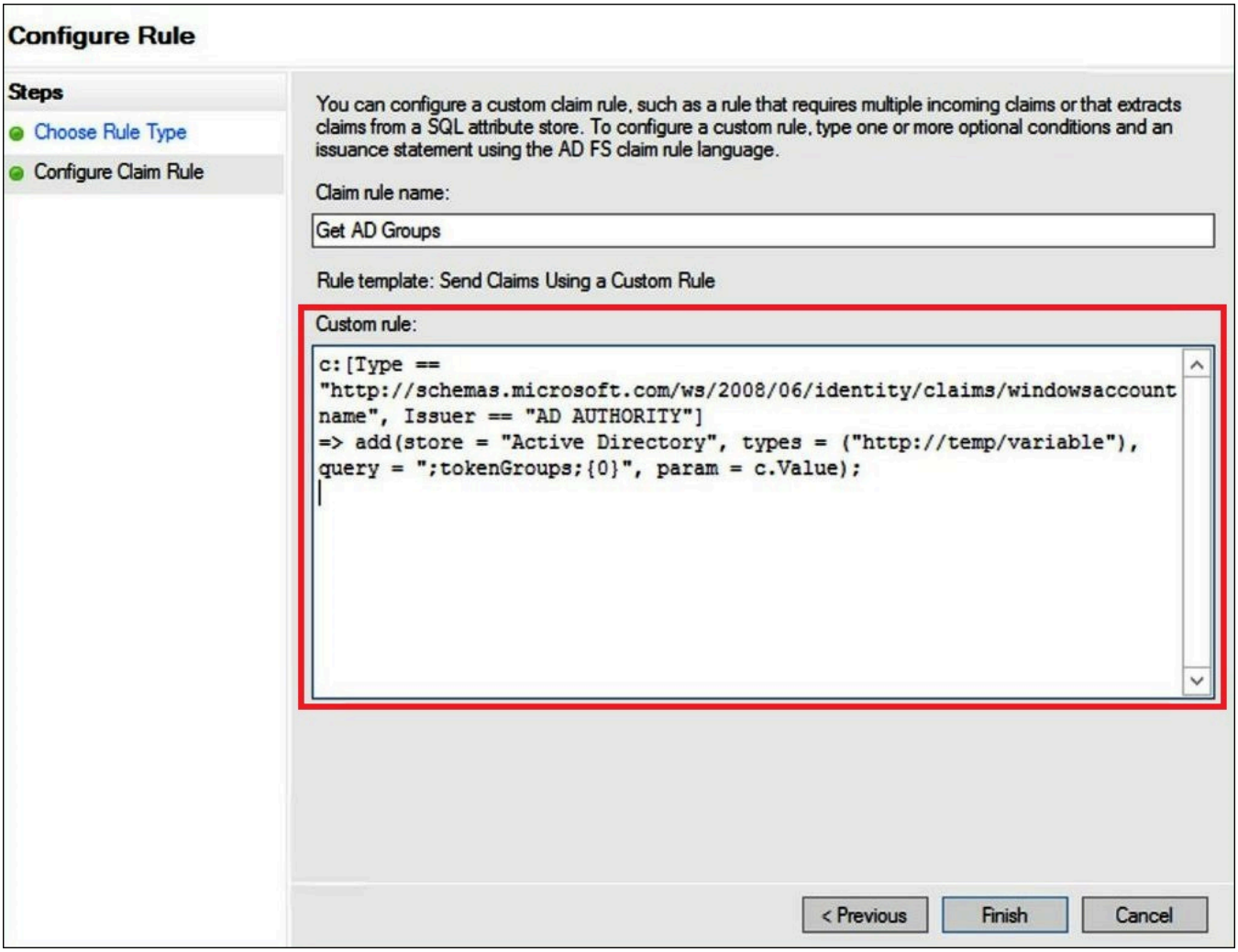
   - Pour Claim rule name (Nom de la règle de revendication), saisissez **`RoleSessionName`**.
   - Pour Rule template (Modèle de règle), utilisez Send LDAP Attribute as Claims (Envoyer les attributs LDAP en tant que revendications).
   - Pour Attribute store (Magasin d'attributs), choisissez Active Directory.
   - Pour Mapping of LDAP attributes to outgoing claim types (Mappage des attributs LDAP aux types de revendications sortantes), ajoutez l'attribut **`E-Mail-Addresses`**. Pour Outgoing Claim Type (Type de revendication sortante), saisissez **`https://aws.amazon.com/SAML/Attributes/RoleSessionName`**.



7. Choisissez Add Rule (Ajouter une règle), saisissez les informations suivantes pour créer la règle de revendication 3, puis choisissez Finish (Terminer).

   - Pour Claim rule name (Nom de la règle de revendication), saisissez **`Get AD Groups`**.
   - Pour Rule template (Modèle de règle), utilisez Send Claims Using a Custom Rule (Envoyer des revendications à l'aide d'une règle personnalisée).
   - Pour Custom rule (Règle personnalisée), saisissez le code suivant :

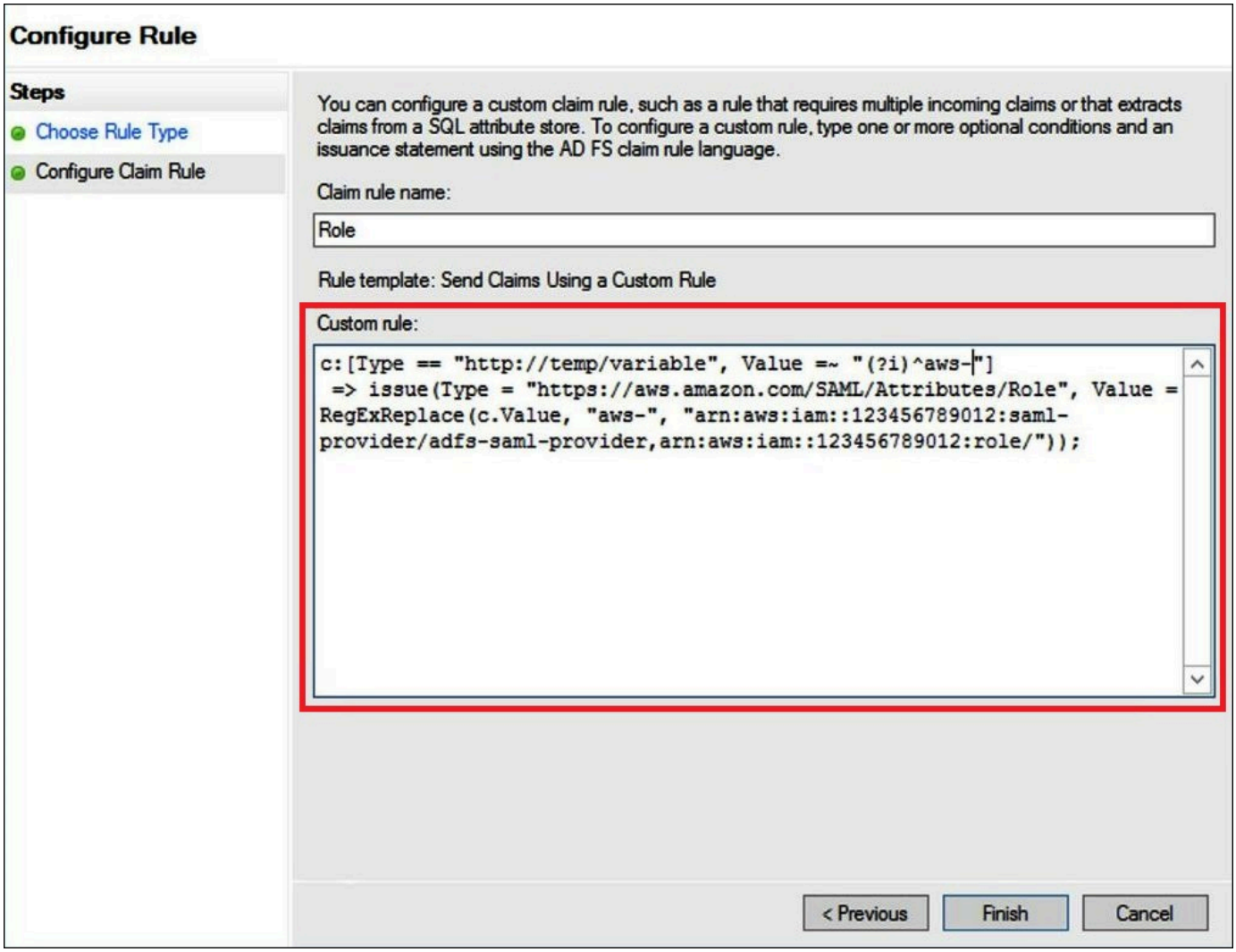
```
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/
windowsaccountname",
 Issuer == "AD AUTHORITY"]=> add(store = "Active Directory", types = ("http://
temp/variable"),
 query = ";tokenGroups;{0}", param = c.Value);
```



8. Choisissez Add Rule (Ajouter une règle). Saisissez les informations suivantes pour créer la règle de revendication 4, puis choisissez Finish (Terminer).

   - Pour Claim rule name (Nom de la règle de revendication), saisissez **`Role`**.
   - Pour Rule template (Modèle de règle), utilisez Send Claims Using a Custom Rule (Envoyer des revendications à l'aide d'une règle personnalisée).
   - Pour Custom rule (Règle personnalisée), saisissez le code suivant avec votre numéro de compte et le nom du fournisseur SAML que vous avez créé précédemment :

```
c:[Type == "http://temp/variable", Value =~ "(?i)^aws-"]=> issue(Type = "https://
aws.amazon.com/SAML/Attributes/Role",
Value = RegExReplace(c.Value, "aws-", "arn:aws:iam::AWS_ACCOUNT_NUMBER:saml-
provider/adfs-saml-provider,arn:aws:iam:: AWS_ACCOUNT_NUMBER:role/"));
```

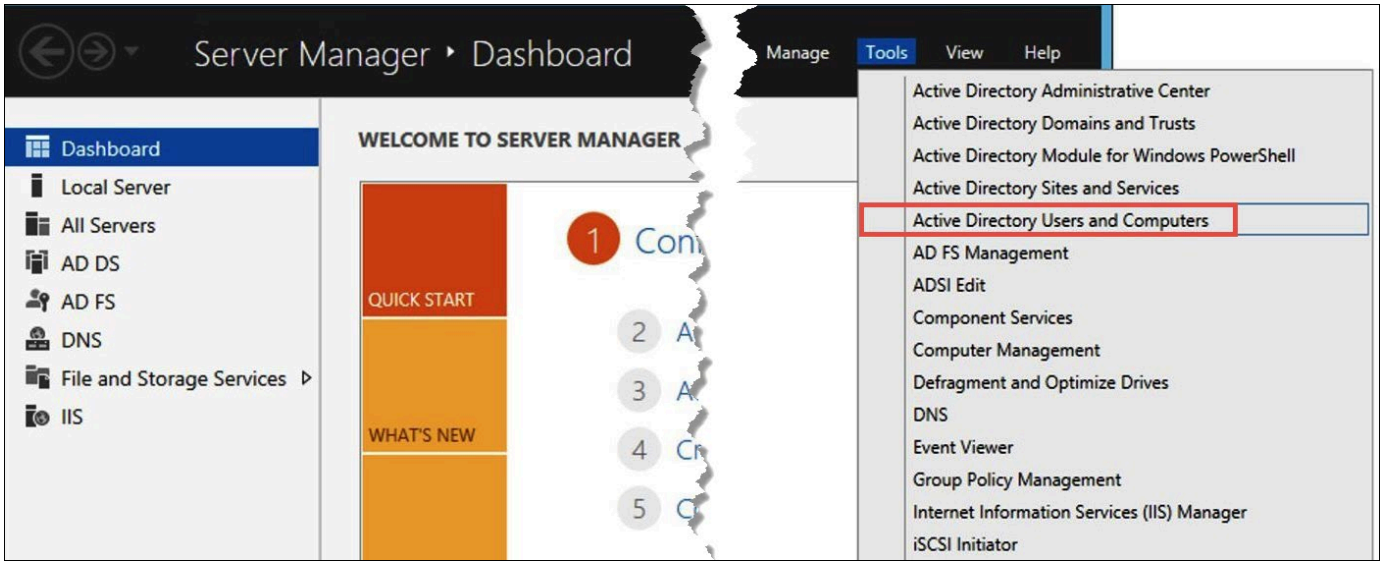


3. Création d'utilisateurs et de groupes Active Directory

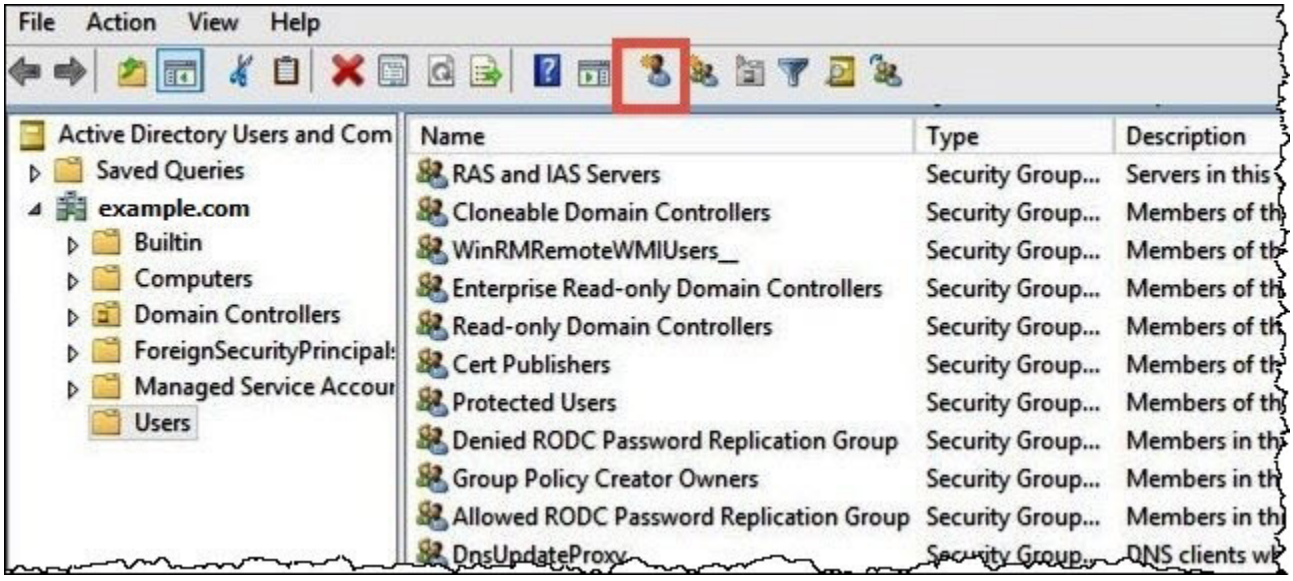
Vous pouvez maintenant créer des utilisateurs AD qui accéderont à Athena, et des groupes AD dans lesquels les placer afin de pouvoir contrôler les niveaux d'accès par groupe. Après avoir créé des groupes AD qui classent les modèles d'accès aux données, vous ajoutez vos utilisateurs à ces groupes.

Créer des utilisateurs AD pour accéder à Athena

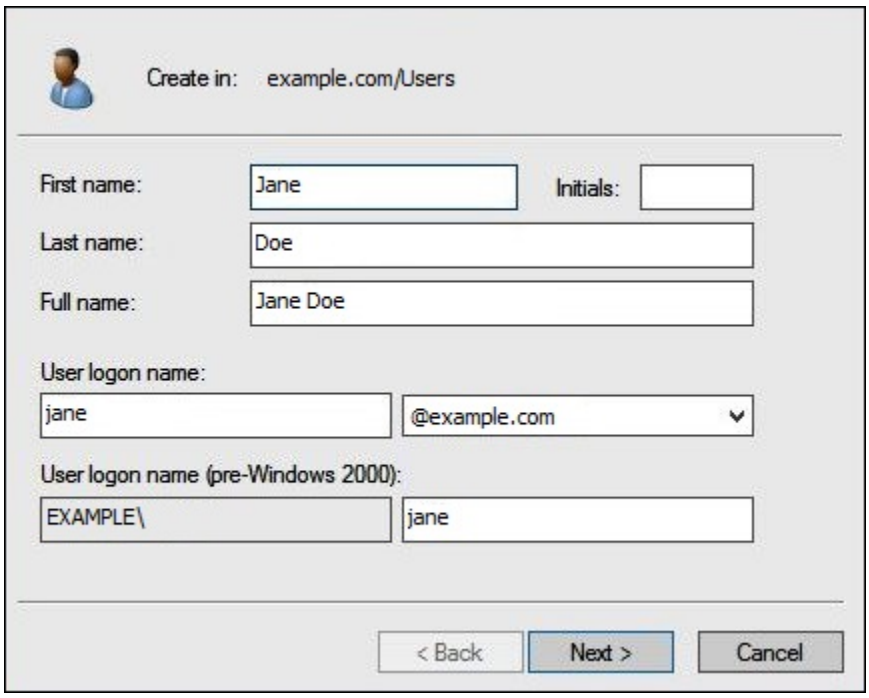
1. Dans le tableau de bord Server Manager (Gestionnaire de serveur), choisissez Tools (Outils), puis choisissez Active Directory Users and Computers (Utilisateurs et ordinateurs Active Directory).



2. Dans le panneau de navigation, choisissez utilisateurs.
3. Dans la barre d'outils Active Directory Users and Computers (Utilisateurs et ordinateurs Active Directory), choisissez l'option Create user (Créer un utilisateur).

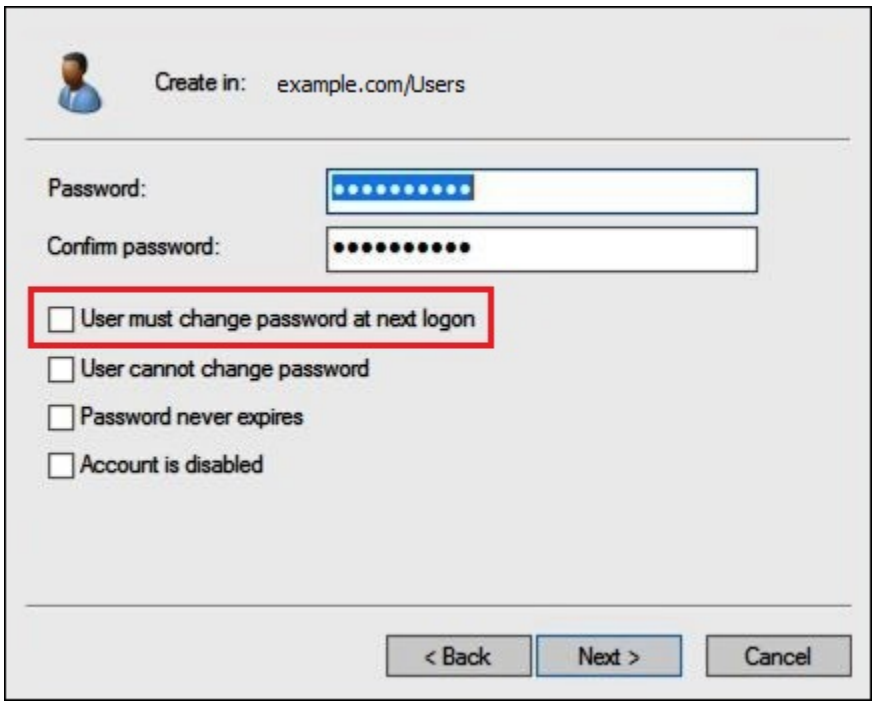


4. Dans la boîte de dialogue New Object – User (Nouvel objet – Utilisateur), saisissez un nom dans les champs First name (Prénom), Last name (Nom de famille) et Full name (Nom complet). Ce tutoriel utilise **`Jane Doe`**.

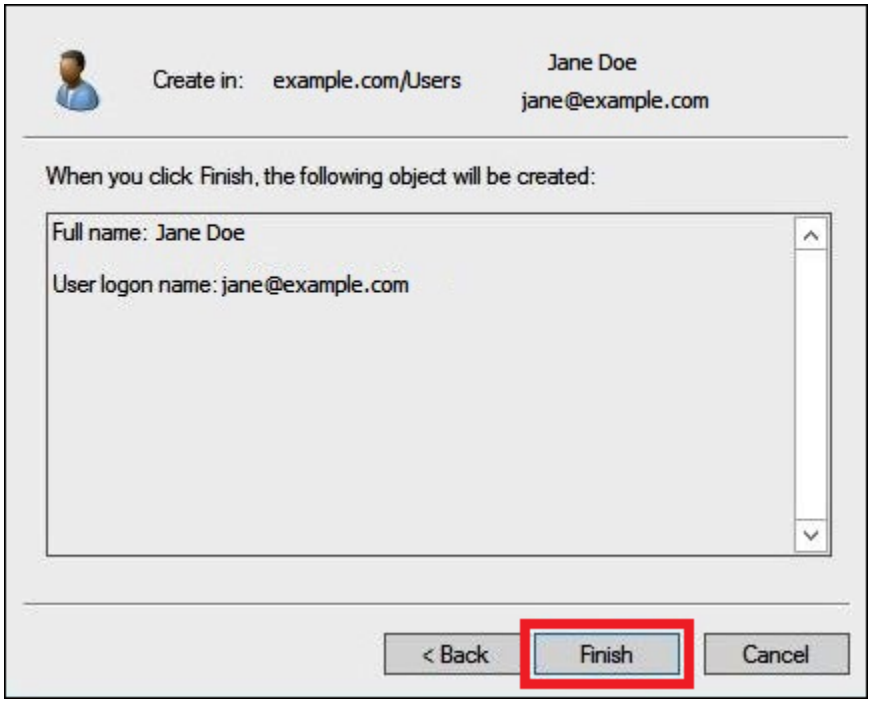


5. Choisissez Suivant.
6. Dans le Password (Mot de passe), saisissez un mot de passe, puis saisissez-le à nouveau pour le confirmer.

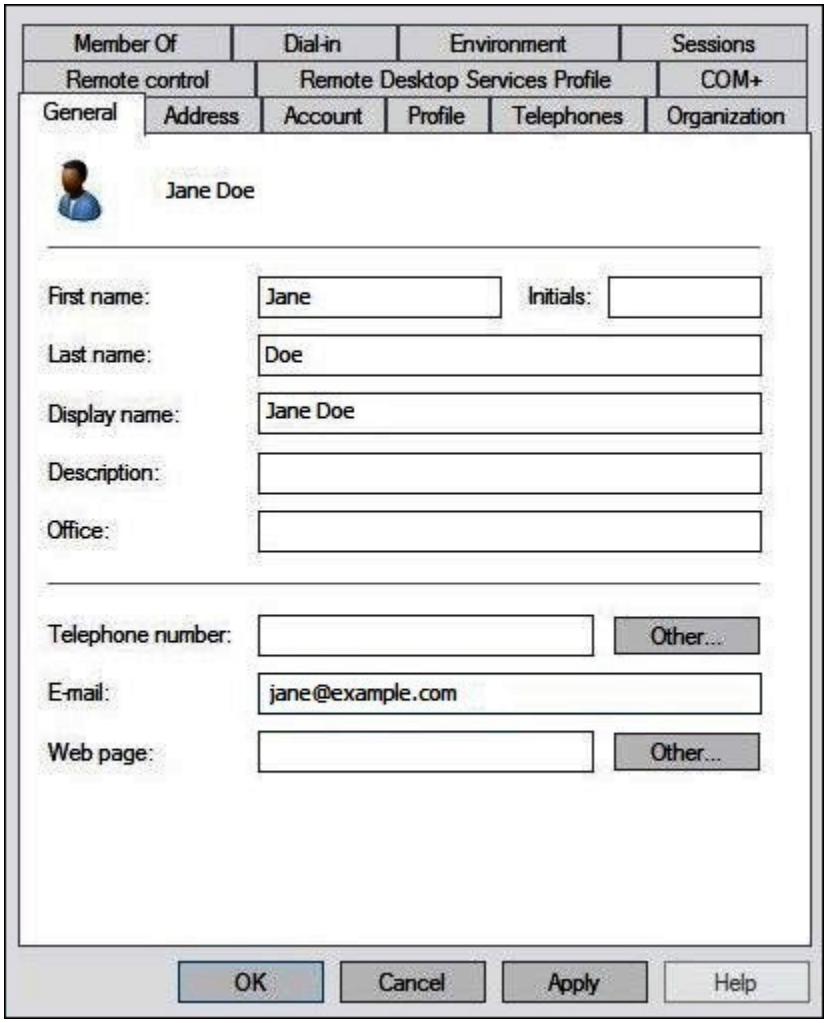
   Pour simplifier, ce tutoriel désélectionne l'option User must change password at next sign on (L'utilisateur doit changer son mot de passe à la prochaine ouverture de session). Dans des scénarios réels, vous devez demander aux utilisateurs nouvellement créés de changer leur mot de passe.



7. Choisissez Suivant.
8. Choisissez Finish (Terminer).



9. Dans Active Directory Users and Computers (Utilisateurs et ordinateurs Active Directory), choisissez le nom de l'utilisateur.
10. Dans la boîte de dialogue Properties (Propriétés) de l'utilisateur, pour E-mail, saisissez une adresse e-mail. Ce tutoriel utilise **`jane@example.com`**.



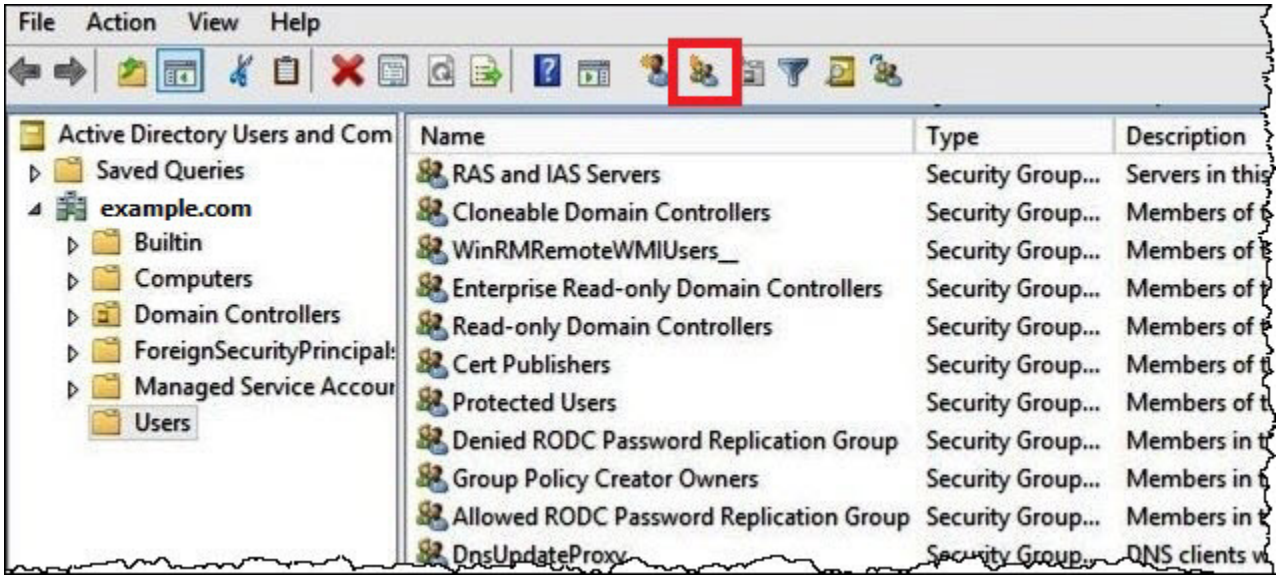
11. Sélectionnez OK.

Création de groupes AD pour représenter les modèles d'accès aux données

Vous pouvez créer des groupes AD dont les membres assument le rôle IAM `adfs-data-access` lorsqu'ils se connectent à AWS. L'exemple suivant crée un groupe AD appelé aws-adfs-data-access.

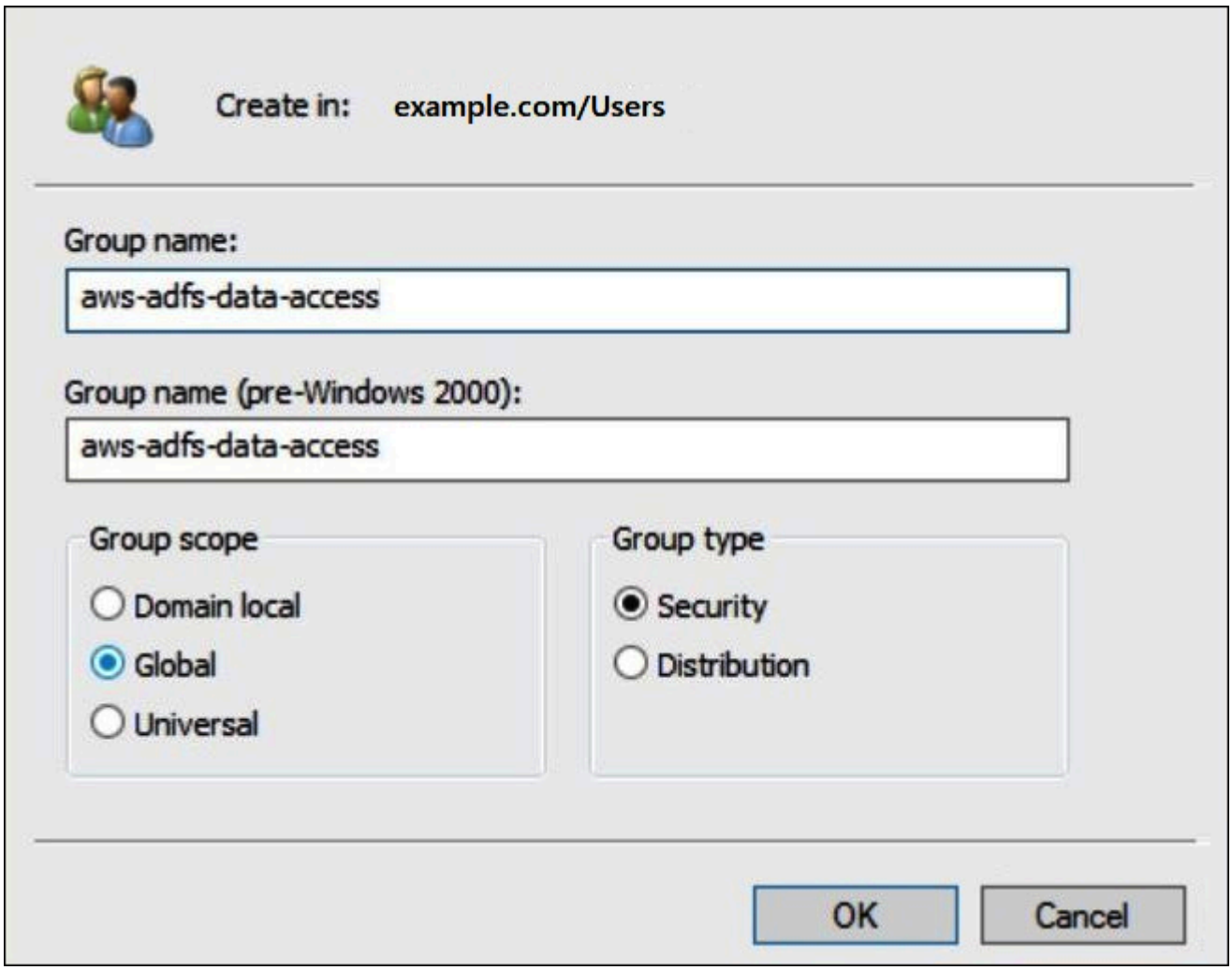
Créer un groupe

1. Dans le tableau de bord Server Manager (Gestionnaire de serveur), dans le menu Tools (Outils), choisissez Active Directory Users and Computers (Utilisateurs et ordinateurs Active Directory).
2. Dans la barre d'outils, choisissez l'option Create new group (Créer un nouveau groupe).



3. Dans la boîte de dialogue New Object – Group (Nouvel objet – Groupe), saisissez les informations suivantes :

   - Pour Group name (Nom du groupe), saisissez **`aws-adfs-data-access`**.
   - Pour Group scope (Étendue du groupe), sélectionnez Global.
   - Pour Group type (Type de groupe), sélectionnez Security (Sécurité).



4. Sélectionnez OK.

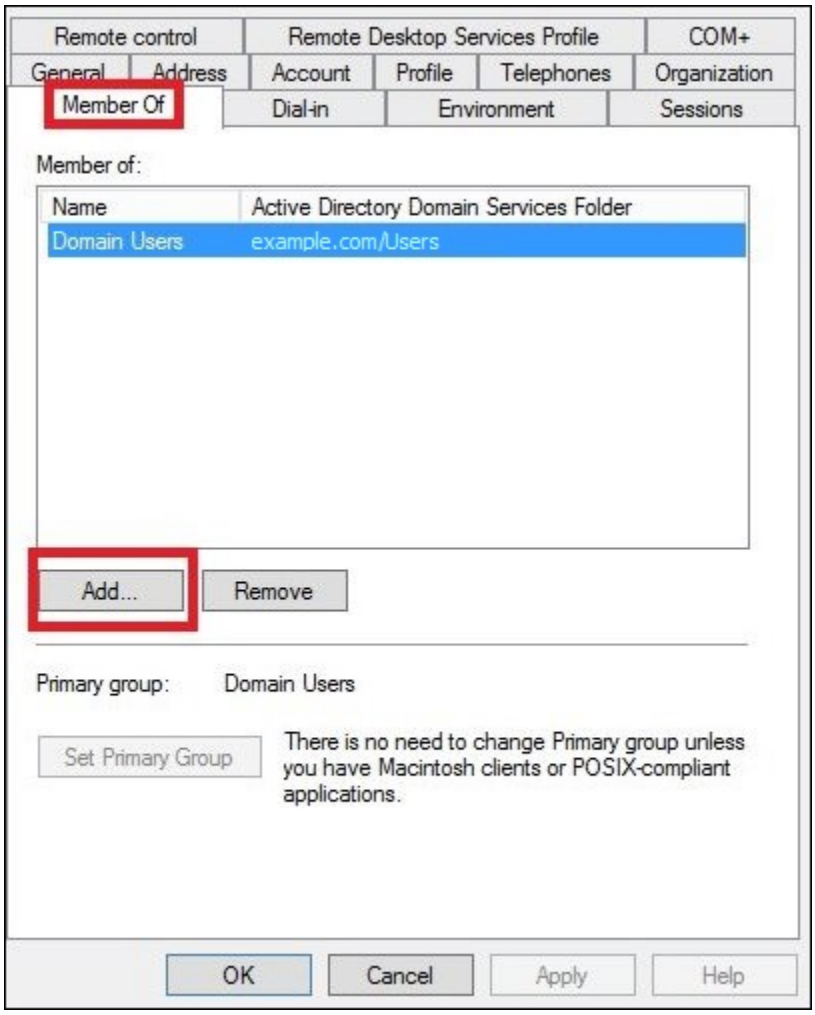
Ajout des utilisateurs AD aux groupes appropriés

Maintenant que vous avez créé un utilisateur AD et un groupe AD, vous pouvez ajouter l'utilisateur au groupe.

Ajouter un utilisateur AD à un groupe AD

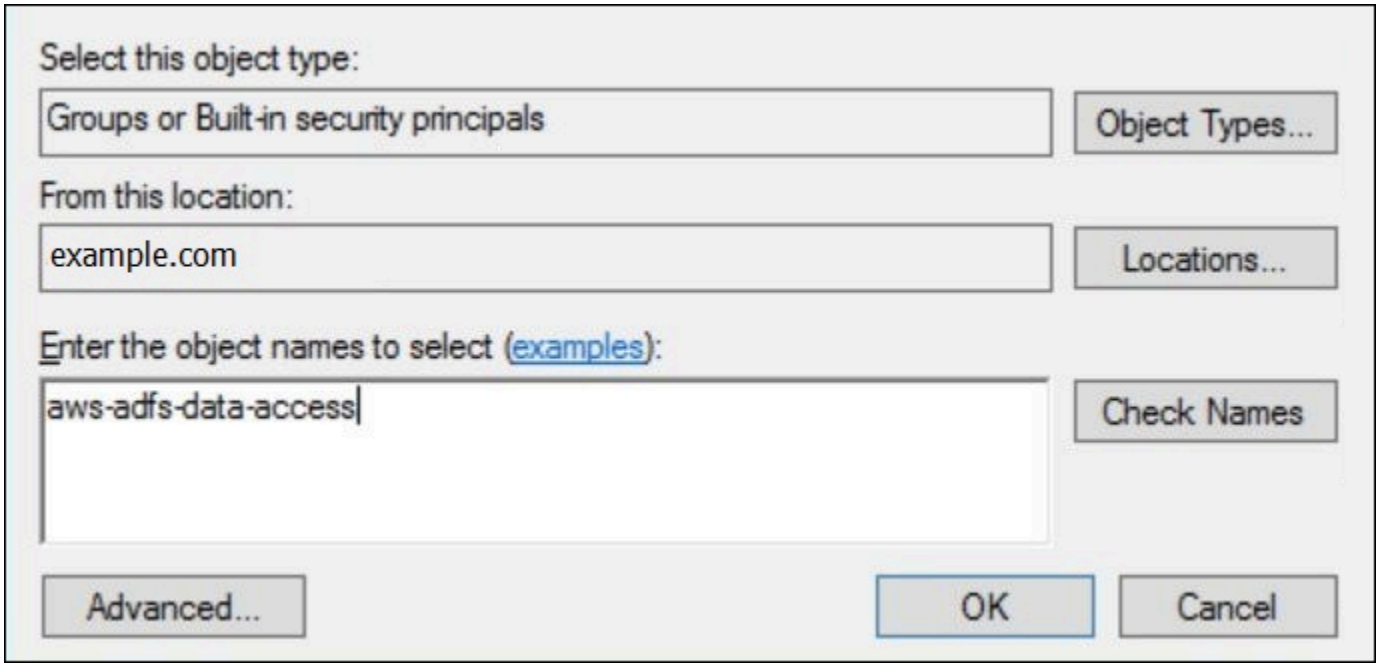
1. Dans le tableau de bord Server Manager (Gestionnaire de serveur), dans le menu Tools (Outils), choisissez Active Directory Users and Computers (Utilisateurs et ordinateurs Active Directory).
2. Pour First name (Prénom) et Last name (Nom de famille), choisissez un utilisateur (par exemple, Jane Doe).

3. Dans la boîte de dialogue Properties (Propriétés) de l'utilisateur, dans l'onglet Member Of (Membre de), choisissez Add (Ajouter).



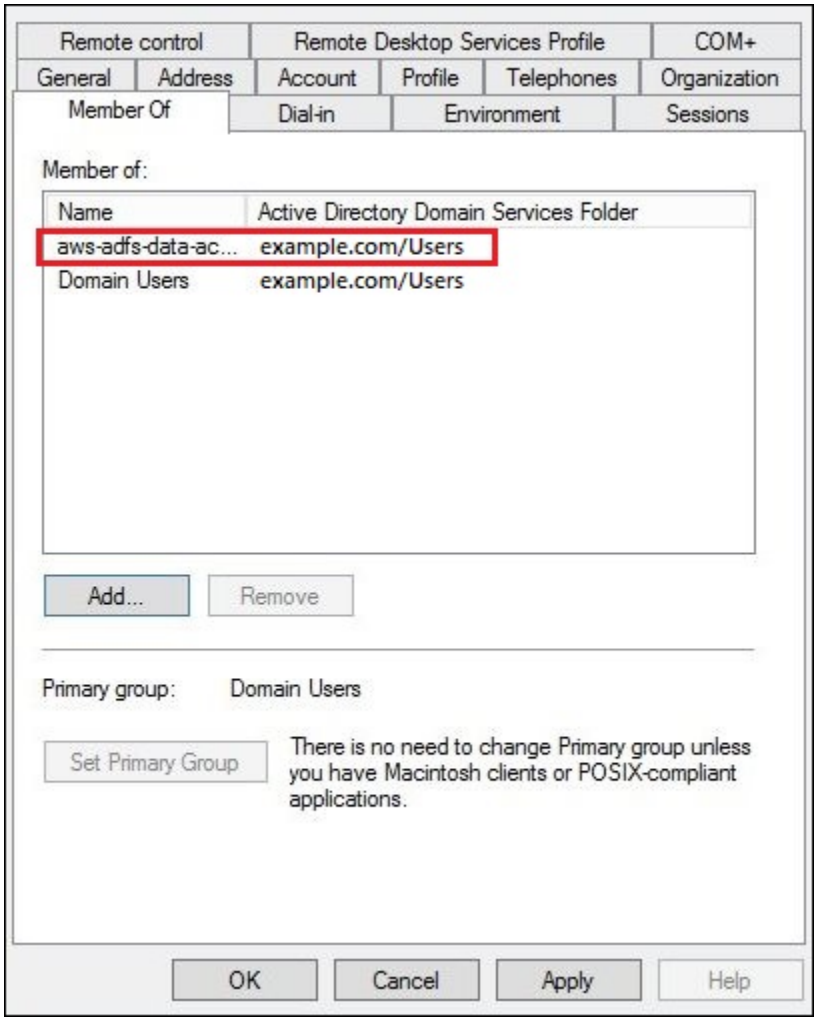
4. Ajoutez un ou plusieurs groupes AD FS en fonction de vos besoins. Ce tutoriel ajoute le groupe aws-adfs-data-access.

5. Dans la boîte de dialogue Select Groups (Sélectionner les groupes), pour Enter the object names to select (Saisir les noms d'objets à sélectionner), saisissez le nom du groupe AD FS que vous avez créé (par exemple, **`aws-adfs-data-access`**), puis choisissez Check Names (Vérifier les noms).



6. Sélectionnez OK.

   Dans la boîte de dialogue Properties (Propriétés) de l'utilisateur, le nom du groupe AD apparaît dans la liste Member of (Membre de).



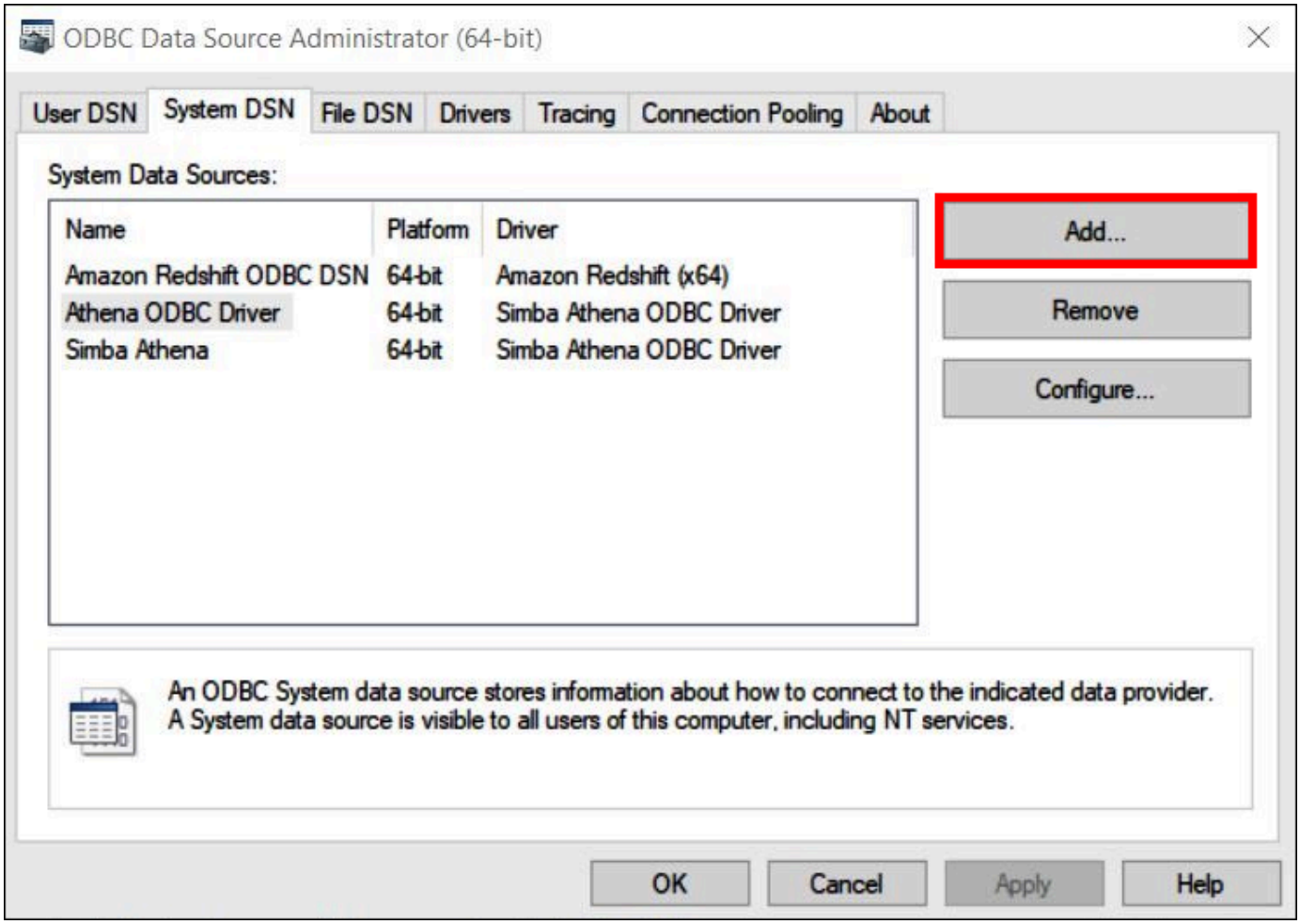
7. Choisissez Apply (Appliquer), puis OK.

4. Configuration de la connexion ODBC AD FS à Athena

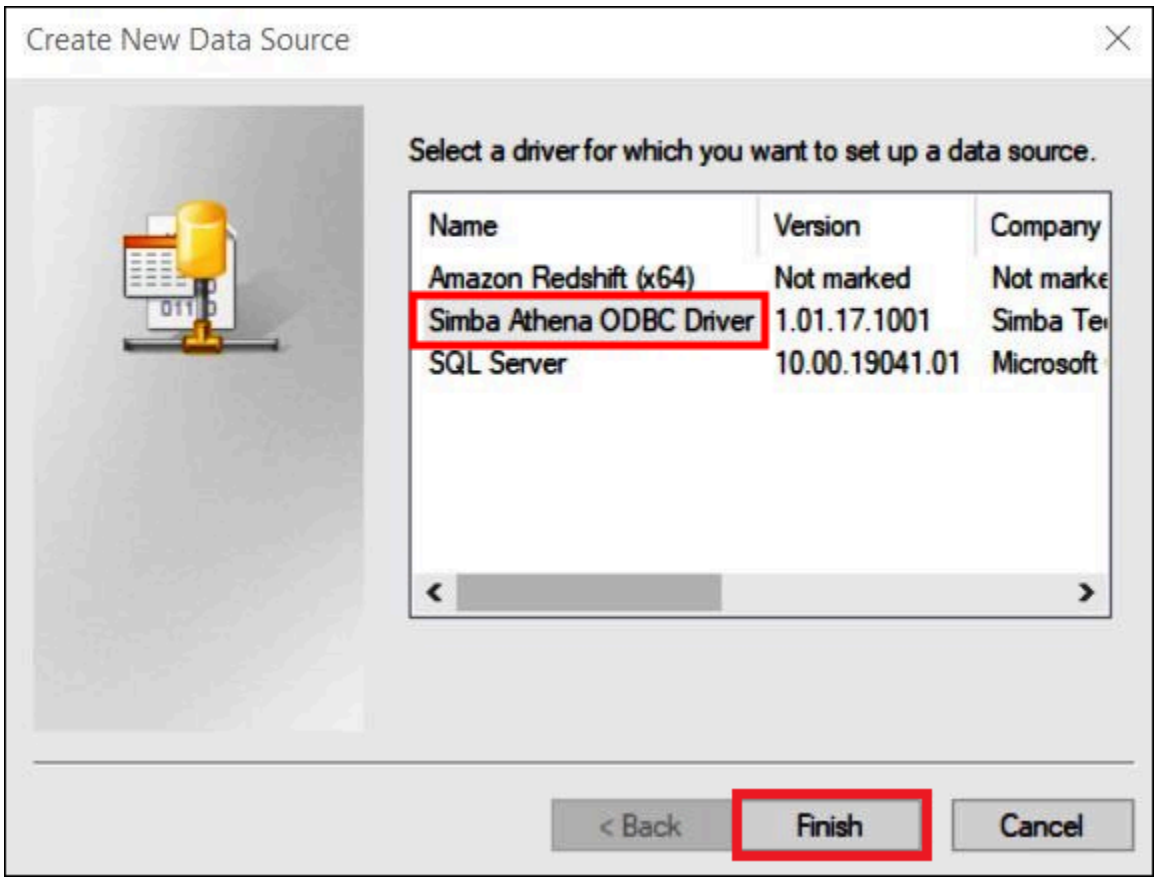
Après avoir créé vos utilisateurs et groupes AD, vous pouvez utiliser le programme Sources de données ODBC de Windows pour configurer votre connexion ODBC Athena pour AD FS.

Configurer la connexion ODBC AD FS à Athena

1. Installez le pilote ODBC pour Athena. Pour les liens de téléchargement, consultez Connexion à Amazon Athena avec le pilote ODBC.
2. Dans Windows, choisissez Démarrer, Sources de données ODBC.
3. Dans le programme Administrateur de source de données ODBC, choisissez Ajouter.

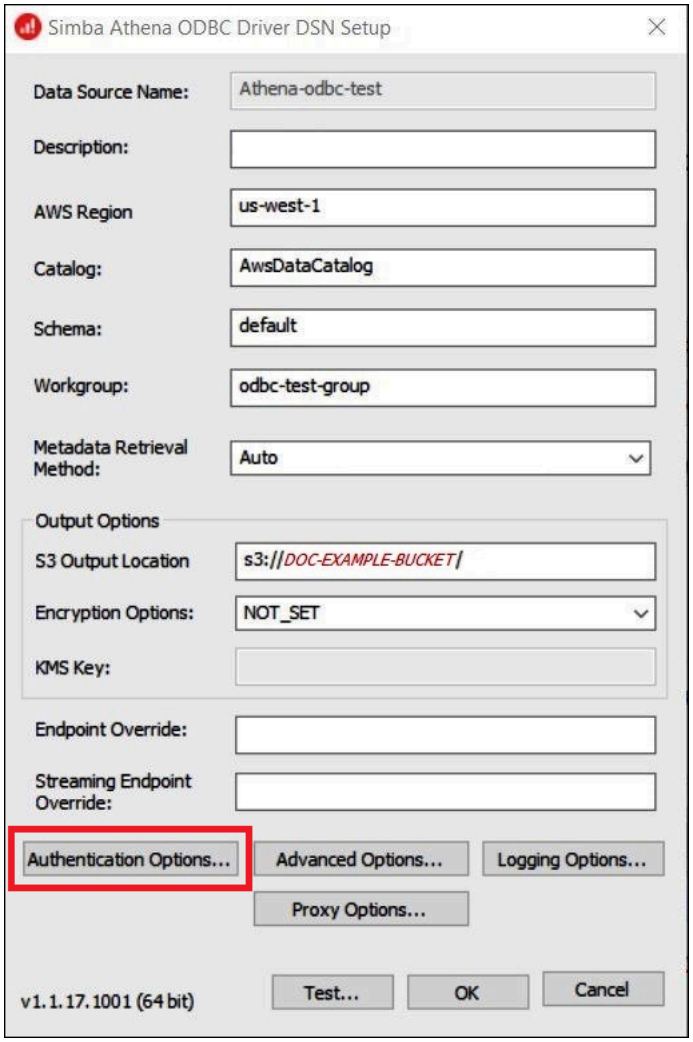


4. Dans la boîte de dialogue Create New Data Source (Créer une nouvelle source de données), choisissez Simba Athena ODBC Driver (Pilote ODBC Simba Athena), puis Finish (Terminer).



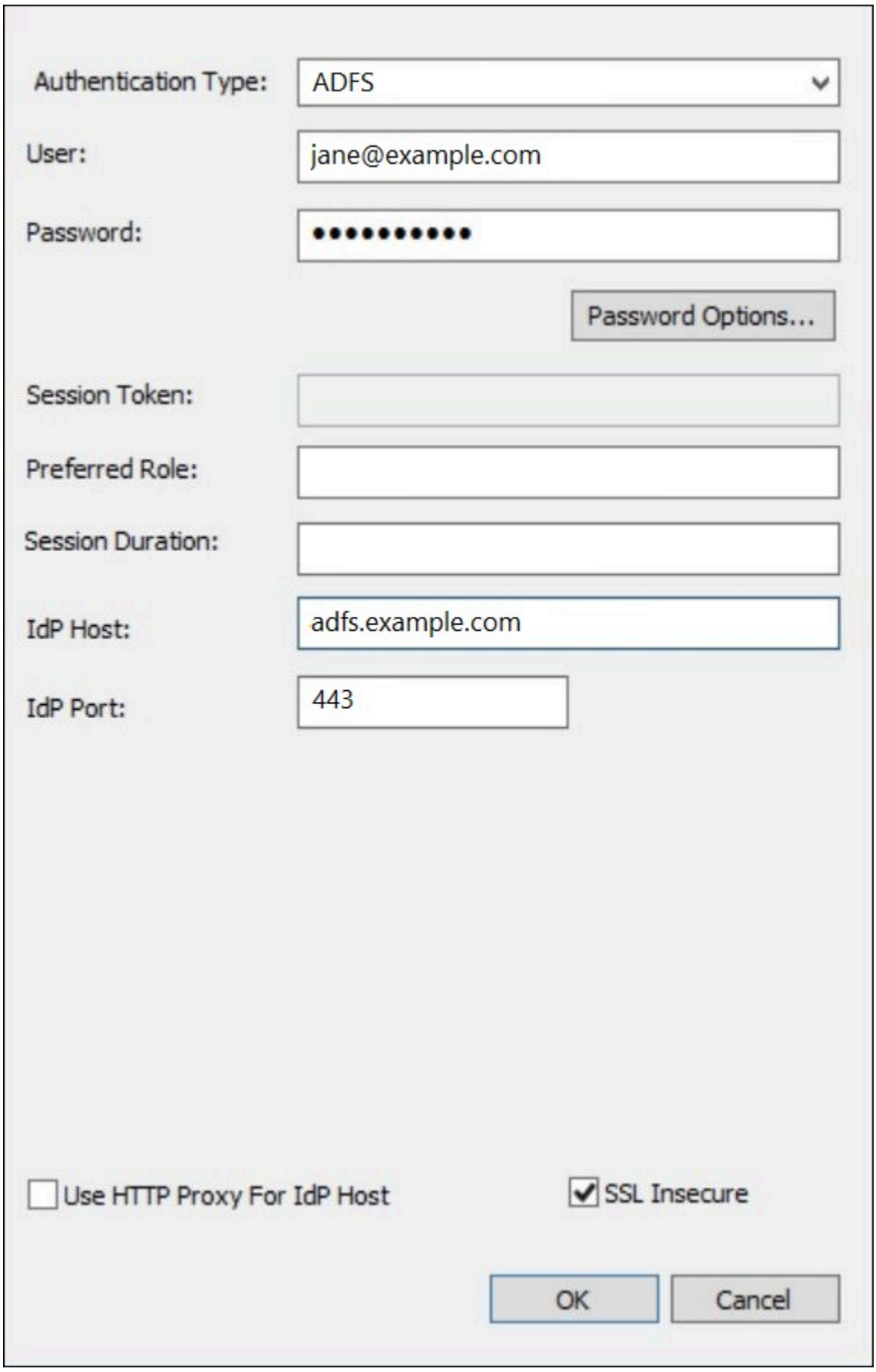
5. Dans la boite de dialogue Simba Athena ODBC Driver DSN Setup (Configuration DSN du pilote ODBC Simba Athena), saisissez les valeurs suivantes :

   - Pour Data Source Name (Nom de la source de données), saisissez un nom pour votre source de données (par exemple, **`Athena-odbc-test`**).
   - Pour Description, saisissez la description de votre source de données.
   - Pour Région AWS, saisissez la Région AWS que vous utilisez (par exemple **`us-west-1`**).
   - Pour Emplacement de sortie S3, saisissez le chemin Amazon S3 où vous souhaitez stocker votre sortie.



6. Choisissez Options d'authentification.
7. Dans la boite de dialogue Authentication Options (Options d'authentification), spécifiez les valeurs suivantes :

   - Pour Authentication Type (Type d'authentification), choisissez ADFS.
   - Pour User (Utilisateur), saisissez l'adresse e-mail de l'utilisateur (par exemple, **`jane@example.com`**).
   - Pour Password (Mot de passe), saisissez le mot de passe ADFS de l'utilisateur.
   - Pour IdP Host (Hôte IdP), saisissez le nom du serveur AD FS (par exemple, **`adfs.example.com`**).
   - Pour IdP Port (Port IdP), utilisez la valeur par défaut 443.
   - Sélectionnez l'option SSL Insecure (SSL non sécurisé).


Authentication Type:
ADFS
User:
jane@example.com
Password:
Password Options...
Session Token:
Preferred Role:
Session Duration:
IdP Host:
adfs.example.com
IdP Port:
443
Use HTTP Proxy For IdP Host
SSL Insecure
OK
Cancel

8. Choisissez OK pour fermer les Authentication Options (Options d'authentification).
9. Choisissez Test (Tester) pour tester la connexion, ou OK pour terminer.

Configuration de SSO pour ODBC en utilisant le plugin Okta et le fournisseur d'identité Okta

Cette page décrit comment configurer le pilote ODBC Amazon Athena et le plugin Okta pour ajouter une fonctionnalité d'authentification unique (SSO) en utilisant le fournisseur d'identité Okta.

Prérequis

Pour suivre les étapes de ce tutoriel, vous devez avoir :

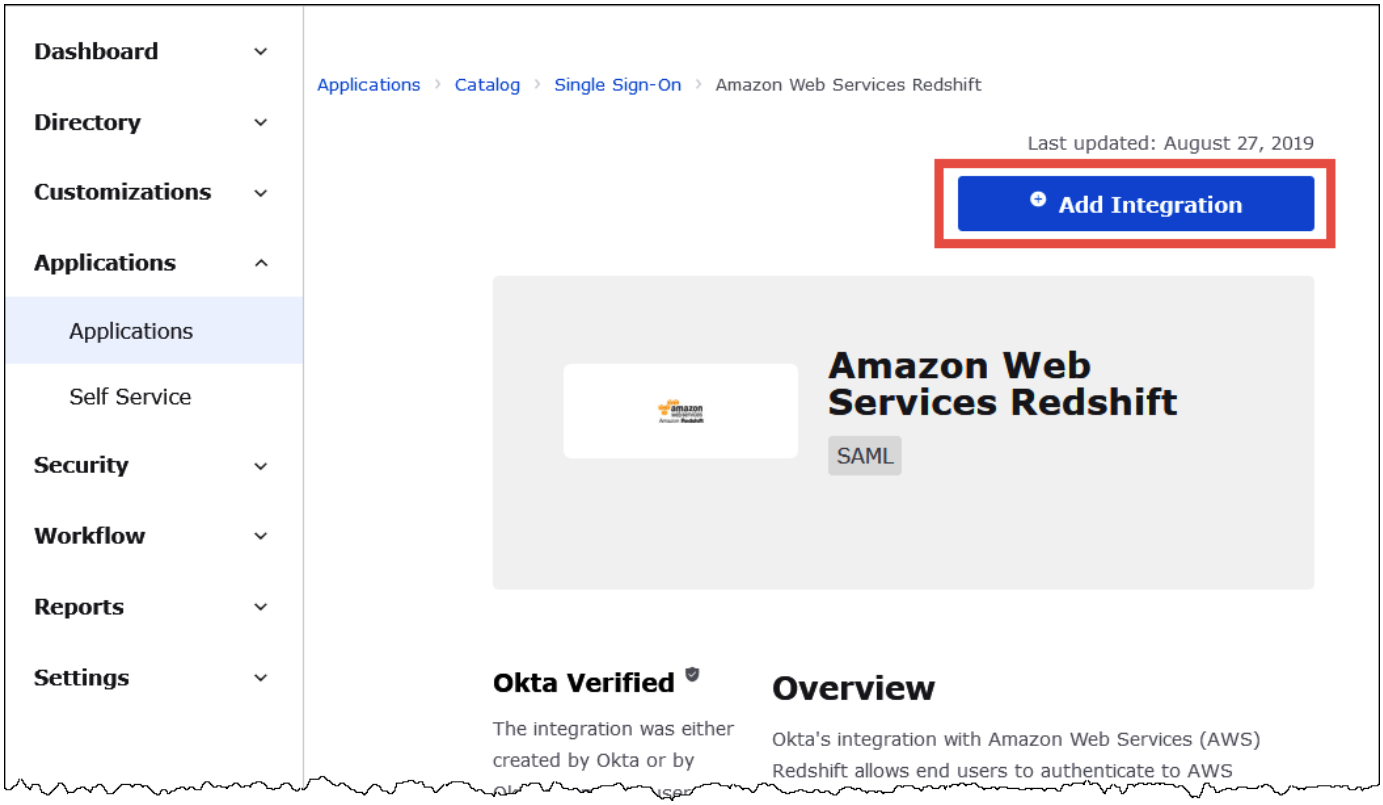
- Pilote ODBC Amazon Athena. Pour les liens de téléchargement, consultez Connexion à Amazon Athena avec le pilote ODBC.
- Un rôle IAM que vous souhaitez utiliser avec SAML. Pour plus d'informations, consultez la section Création d'un rôle pour la fédération SAML 2.0 du Guide de l'utilisateur IAM.
- Un compte Okta. Pour plus d'informations, visitez Okta.com.

Création d'une intégration d'appli dans Okta

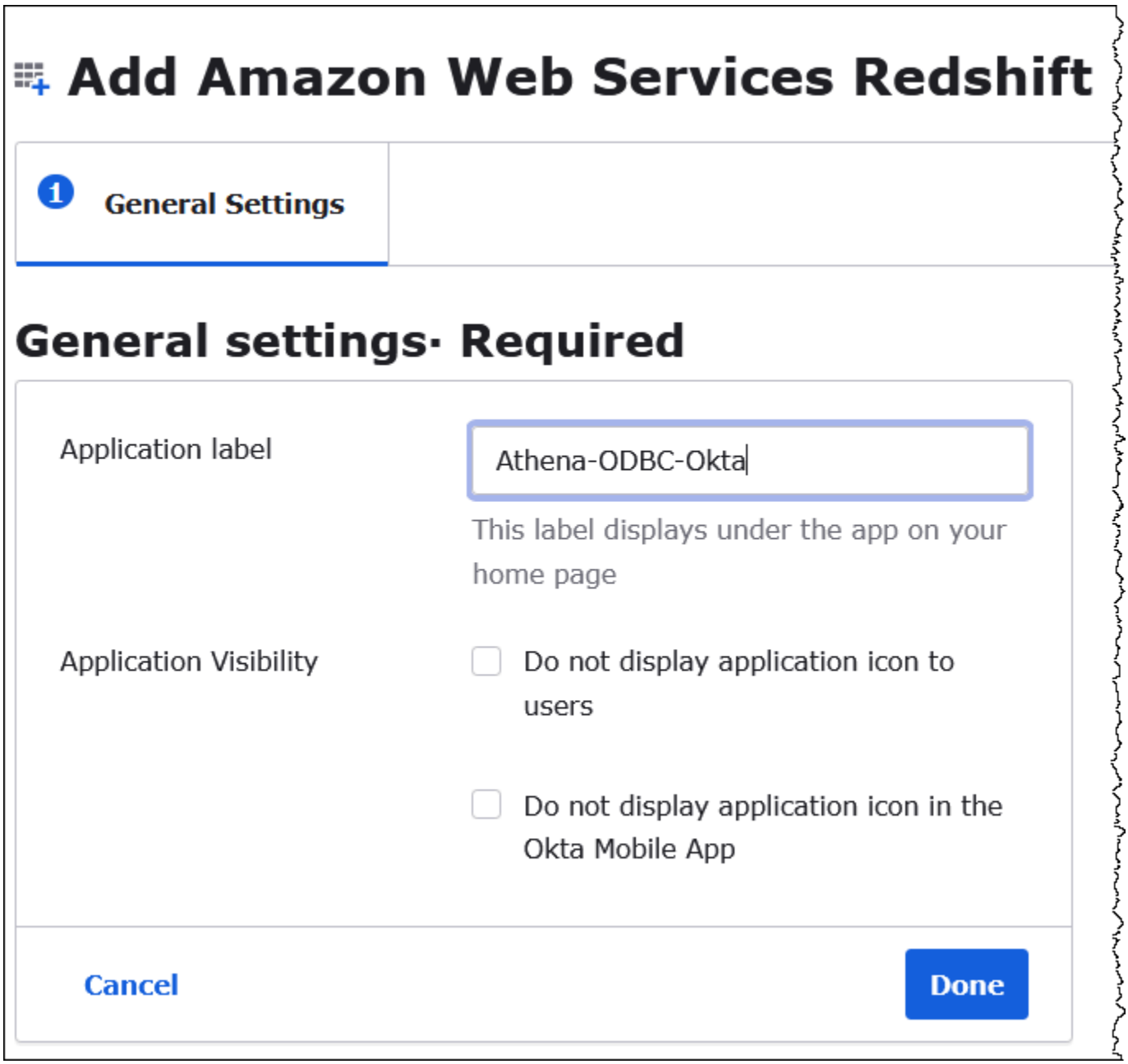
Tout d'abord, utilisez le tableau de bord Okta pour créer et configurer une appli SAML 2.0 pour l'authentification unique à Athena. Vous pouvez utiliser une application Redshift existante dans Okta pour configurer l'accès à Athena.

Pour créer une intégration d'application dans Okta

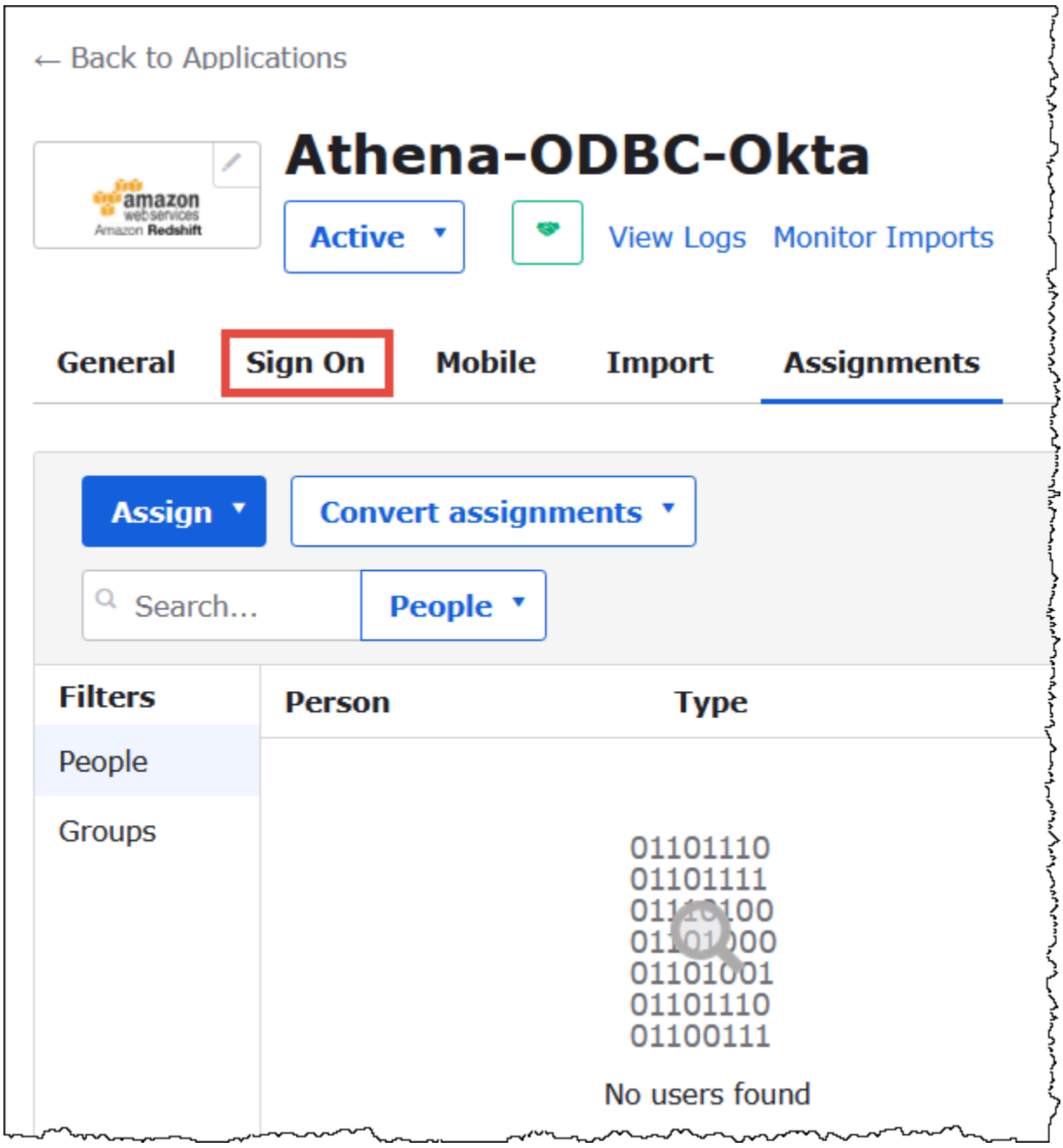
1. Connectez-vous à la page d'administration de votre compte sur Okta.com.
2. Dans le panneau de navigation, choisissez Applications, Applications.
3. Sur la page Applications, choisissez Browse App Catalog (Parcourir le catalogue d'applications).
4. Sur la page Browse App Integration Catalog (Parcourir le catalogue d'intégration d'applications), dans la section Use Case (Cas d'utilisation), choisissez All Integrations (Toutes les intégrations).
5. Dans le champ de recherche, saisissez Amazon Web Services Redshift, puis choisissez Amazon Web Services Redshift SAML.
6. Choisissez Add Integration (Ajouter une intégration).



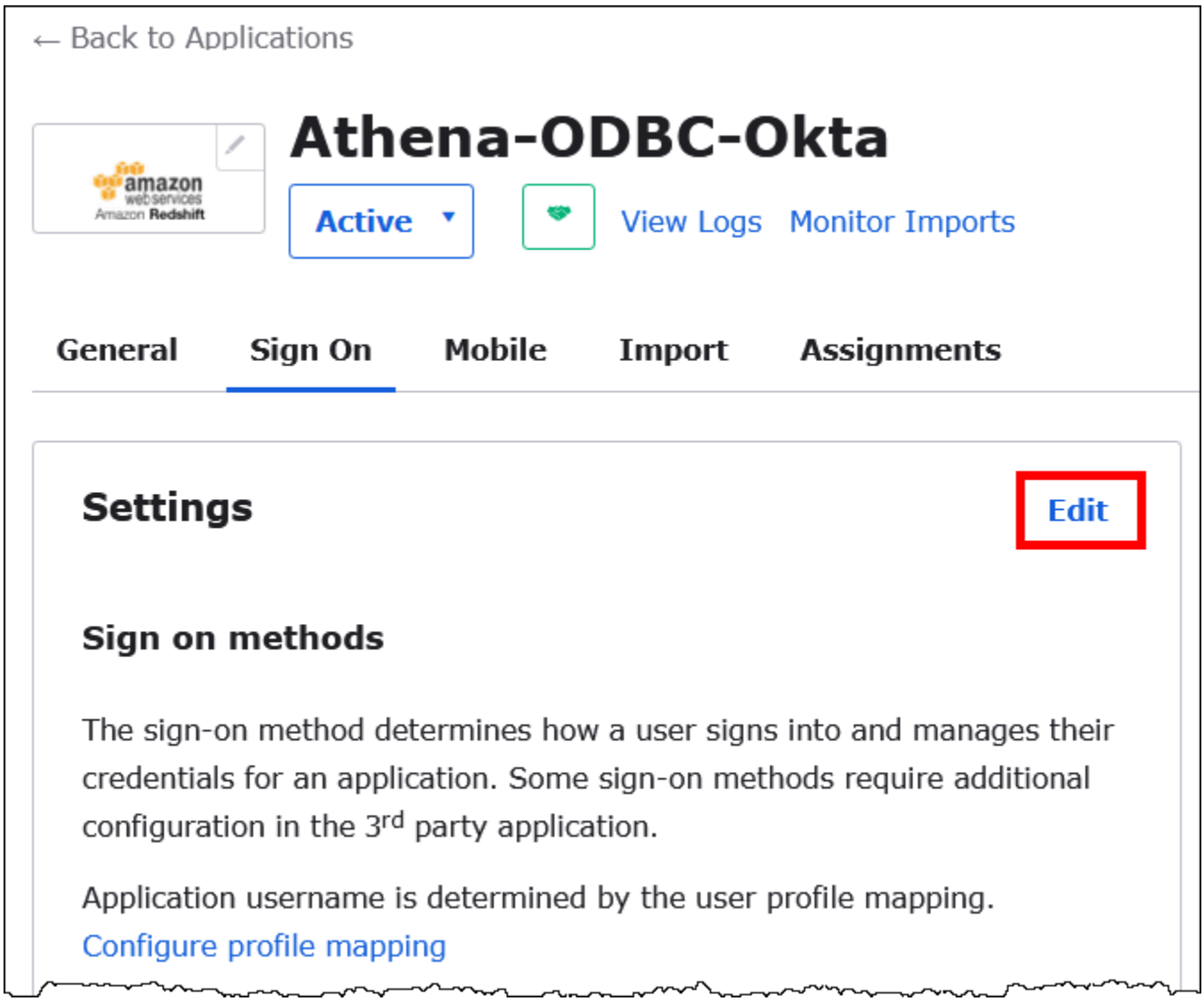
7. Dans la section General Settings Required (Paramètres généraux requis), pour Application label (Étiquette d'application), saisissez un nom pour l'application. Ce tutoriel utilise le nom Athena-ODBC-Okta.



8. Sélectionnez Exécuté.
9. Sur la page de votre application Okta (par exemple, Athena-ODBC-Okta), choisissez Sign On (Se connecter).



10. Dans la section Settings (Paramètres), choisissez Edit (Modifier).



11. Dans la section Advanced Sign-on Settings (Paramètres de connexion avancés), configurez les valeurs suivantes.

    - Pour l'ARN IdP et l'ARN du rôle, entrez votre ARN AWS IDP et votre ARN du rôle sous forme de valeurs séparées par des virgules. Pour plus d'informations sur le format des rôles IAM, consultez la section Configuration des assertions SAML pour la réponse d'authentification dans le Guide de l'utilisateur IAM.
    - Pour Session Duration (Durée de la session), saisissez une valeur comprise entre 900 et 43 200 secondes. Ce tutoriel utilise la valeur par défaut de 3 600 (1 heure).

Advanced Sign-on Settings

These fields may be required for a Amazon Web Services Redshift proprietary sign-on option or general setting.

Idp ARN and Role ARN	arn:aws:iam::1234567890:saml-provid Enter your AWS IDP ARN and Role ARN as comma separated values (e.g. "arn:aws:iam::1234567890:saml-provider/OKTA,arn:aws:iam::1234567890:role/SAML_ROLE").
Session Duration	3600 Set the user's session duration in seconds here. Valid range is 900 to 43200.
DB User Format (Redshift)	${user.username} EL expression to get DB User value (e.g. "${user.username}", "${user.firstName}${user.lastName}@acme.com")
Auto Create (Redshift)	☑ AutoCreate Redshift property (Create a new database user if one does not exist)
Allowed DB Groups (Redshift)	 Comma separated list of allowed user groups. Use "*" to allow all groups, "\," to escape comma in group name

Les paramètres DbUser Format et AutoCreateAllowed DBGroups ne sont pas utilisés par Athena. Vous n'avez pas besoin de les configurer.

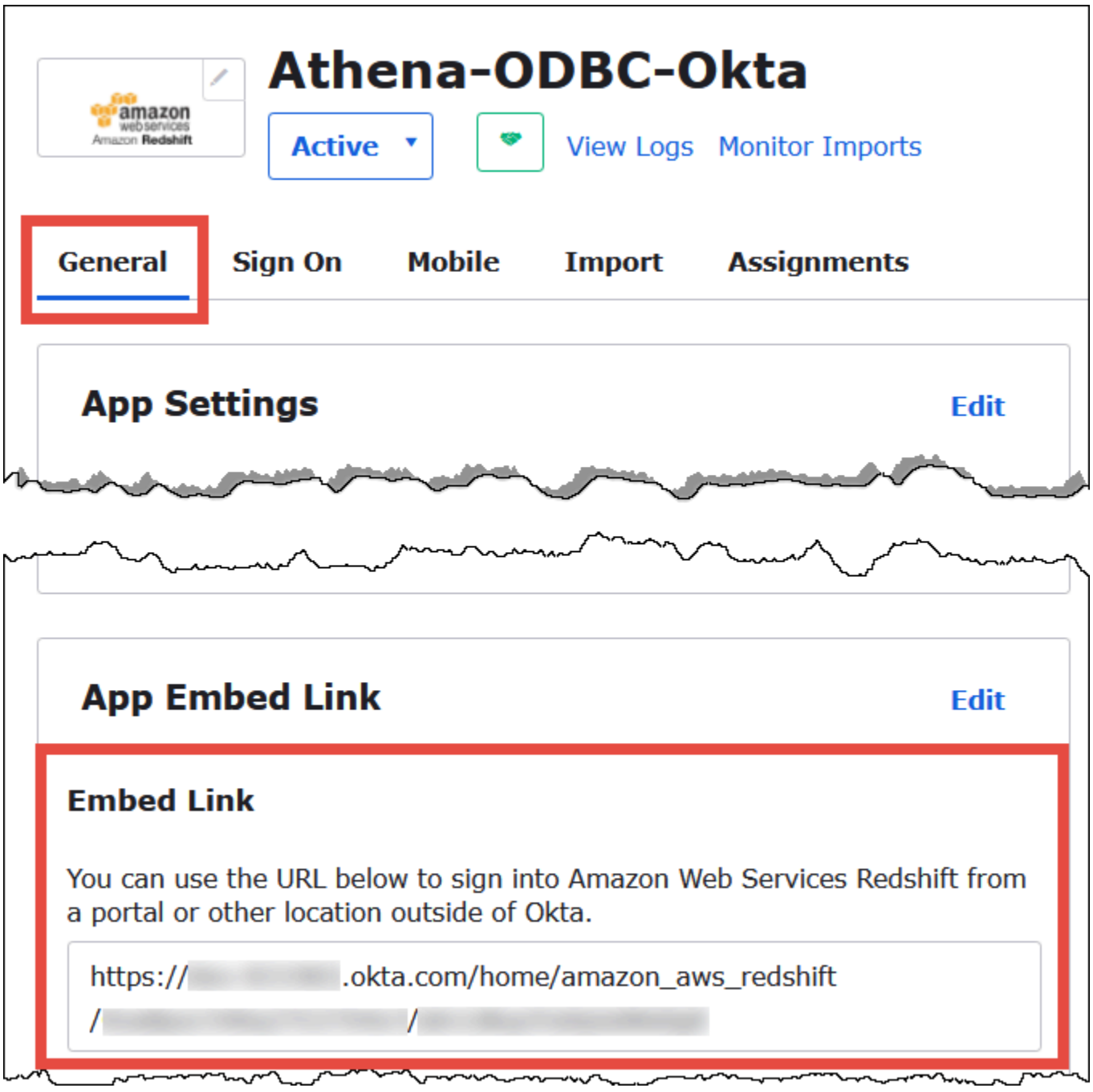
12. Choisissez Enregistrer.

Récupération des informations de configuration ODBC d'Okta

Maintenant que vous avez créé l'application Okta, vous êtes prêt à récupérer l'ID de l'application et l'URL de l'hôte IdP. Vous en aurez besoin plus tard lorsque vous configurerez ODBC pour la connexion à Athena.

Pour récupérer les informations de configuration pour ODBC depuis Okta

1. Choisissez l'onglet General (Général) de votre application Okta, puis faites défiler vers le bas jusqu'à la section App Embed Link (Lien intégré de l'application).



Votre URL d'Embed Link (Lien intégré) est au format suivant :

```
https://trial-1234567.okta.com/home/amazon_aws_redshift/Abc1de2fghi3J45kL678/
abc1defghij2klmNo3p4
```

2. A partir de votre URL d'Embed Link (Lien intégré), extrayez et enregistrez les morceaux suivants :

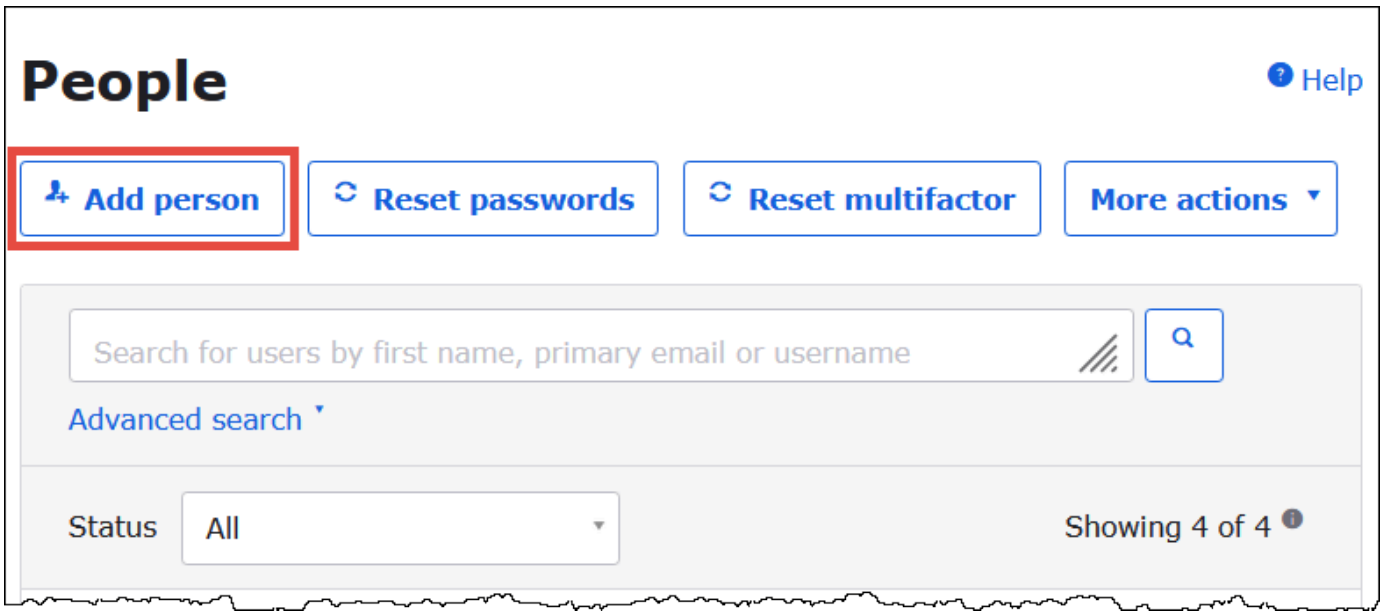
- Le premier segment après `https://`, jusqu'à et y compris `okta.com` (par exemple, trial-1234567.okta.com). Il s'agit de votre hôte IdP.
- Les deux derniers segments de l'URL, y compris la barre oblique au milieu. Les segments sont deux chaînes de 20 caractères avec un mélange de chiffres et de lettres majuscules et minuscules (par exemple, Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4). Il s'agit de l'ID de votre application.

Ajout d'un utilisateur à l'application Okta

Vous êtes maintenant prêt à ajouter un utilisateur à votre application Okta.

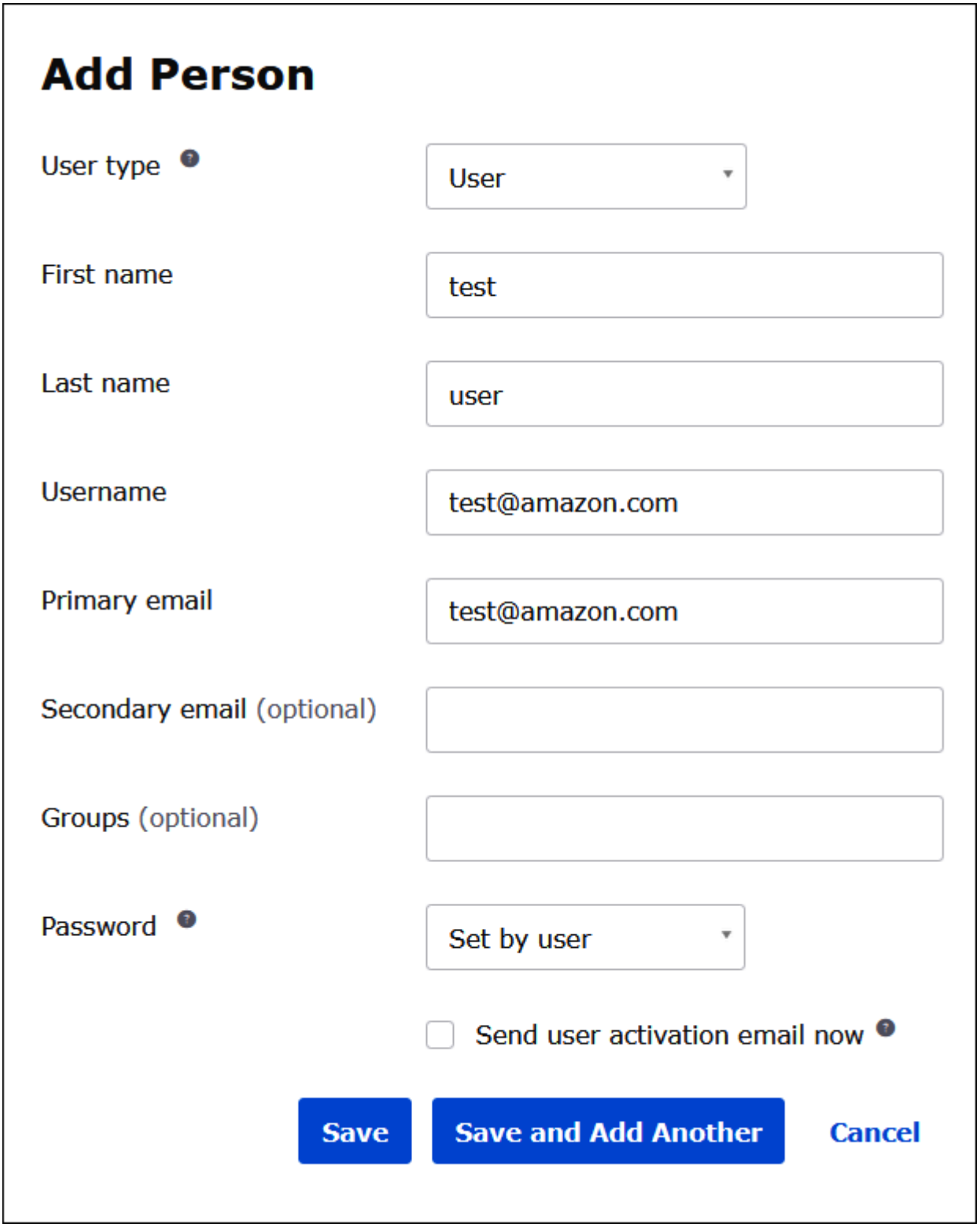
Pour ajouter un utilisateur à l'application Okta

1. Dans le panneau de navigation de gauche, choisissez Directory (Répertoire), puis choisissez People (Personnes).
2. Choisissez Add person (Ajouter une personne).



3. Dans la boîte de dialogue Add Person (Ajouter une personne), saisissez les informations suivantes.
   - Saisissez les valeurs pour First name (Prénom) et Last name (Nom de famille). Ce tutoriel utilise **`test user`**.

- Saisissez des valeurs pour Username (Nom d'utilisateur) et Primary email (E-mail principal). Ce tutoriel utilise **`test@amazon.com`** pour les deux. Vos exigences en matière de sécurité des mots de passe peuvent varier.

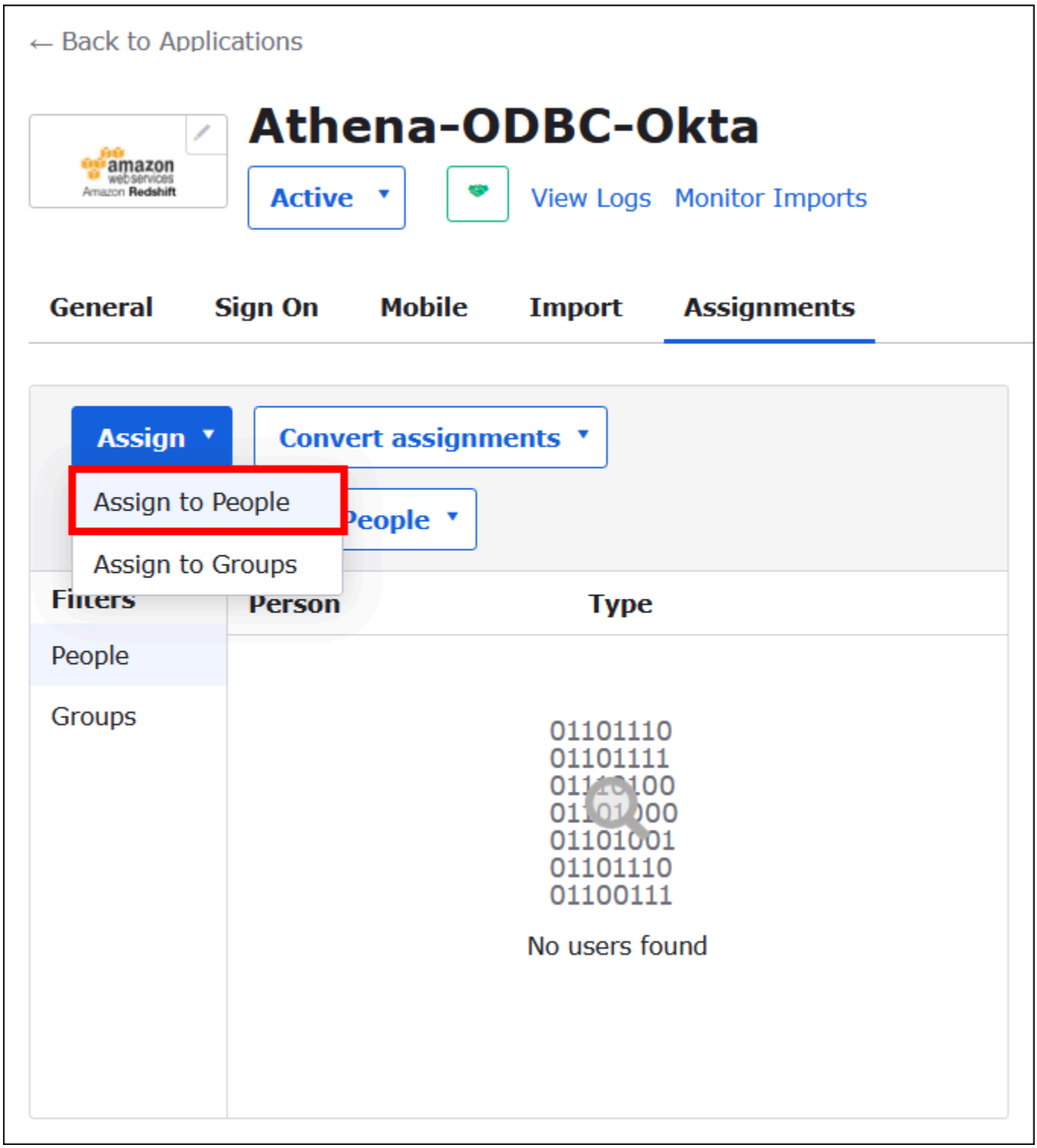


4. Choisissez Enregistrer.

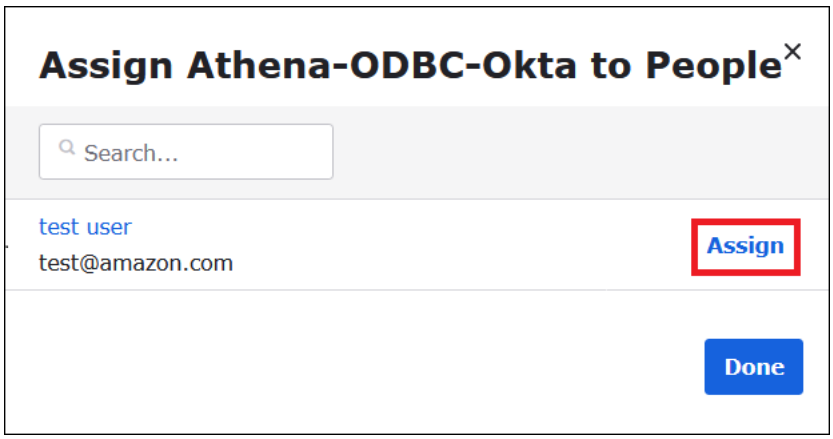
Vous êtes maintenant prêt à affecter l'utilisateur que vous avez créé à votre application.

Pour affecter l'utilisateur à votre application :

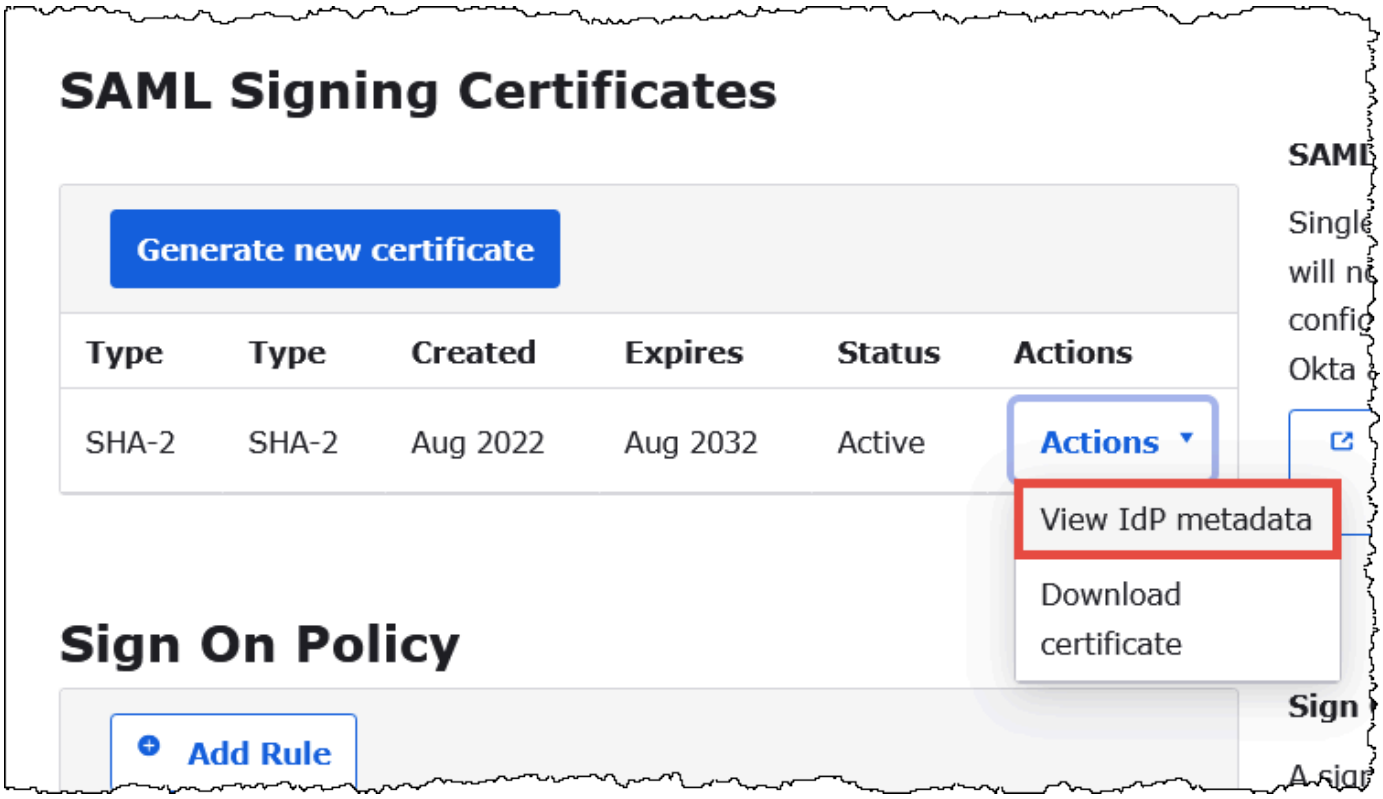
1. Dans le panneau de navigation, choisissez Applications, Applications, puis choisissez le nom de votre application (par exemple, Athena-ODBC-Okta).
2. Choisissez Assign (Affecter), puis Assign to People (Affecter à des personnes).



3. Choisissez l'option Assign (Affecter) pour votre utilisateur, puis cliquez sur Done (Terminé).



4. À l'invite, choisissez Save and Go Back (Sauvegarder et revenir). La boîte de dialogue indique que le statut de l'utilisateur est Assigned (Affecté).
5. Sélectionnez Exécuté.
6. Choisissez l'onglet Sign On (Se connecter).
7. Faites défiler vers le bas jusqu'à la section SAML Signing Certificates (Certificats de signature SAML).
8. Choisissez Actions.
9. Ouvrez le menu contextuel (clic droit) pour View IdP metadata (Afficher les métadonnées IdP), puis choisissez l'option du navigateur pour enregistrer le fichier.
10. Enregistrez le fichier avec une extension `.xml`.

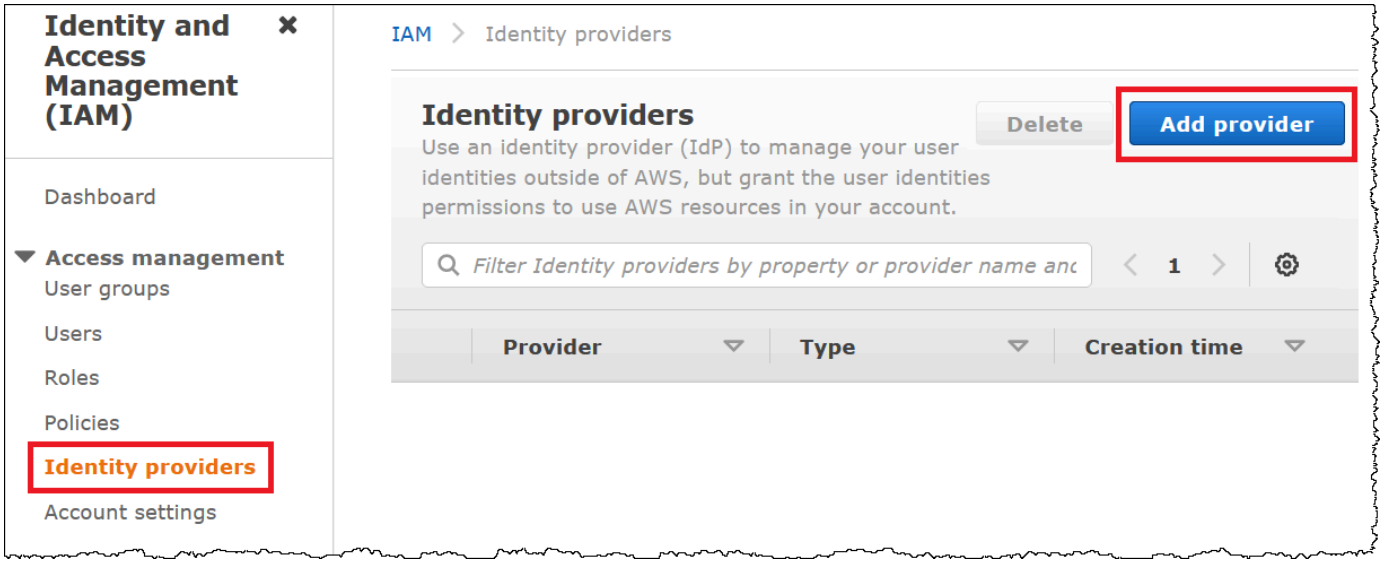


Création d'un fournisseur d'identité et d'un rôle AWS SAML

Vous êtes maintenant prêt à télécharger le fichier XML de métadonnées sur la console IAM dans AWS. Vous allez utiliser ce fichier pour créer un fournisseur d'identité et un rôle AWS SAML. Utilisez un compte d'administrateur des services AWS pour effectuer ces étapes.

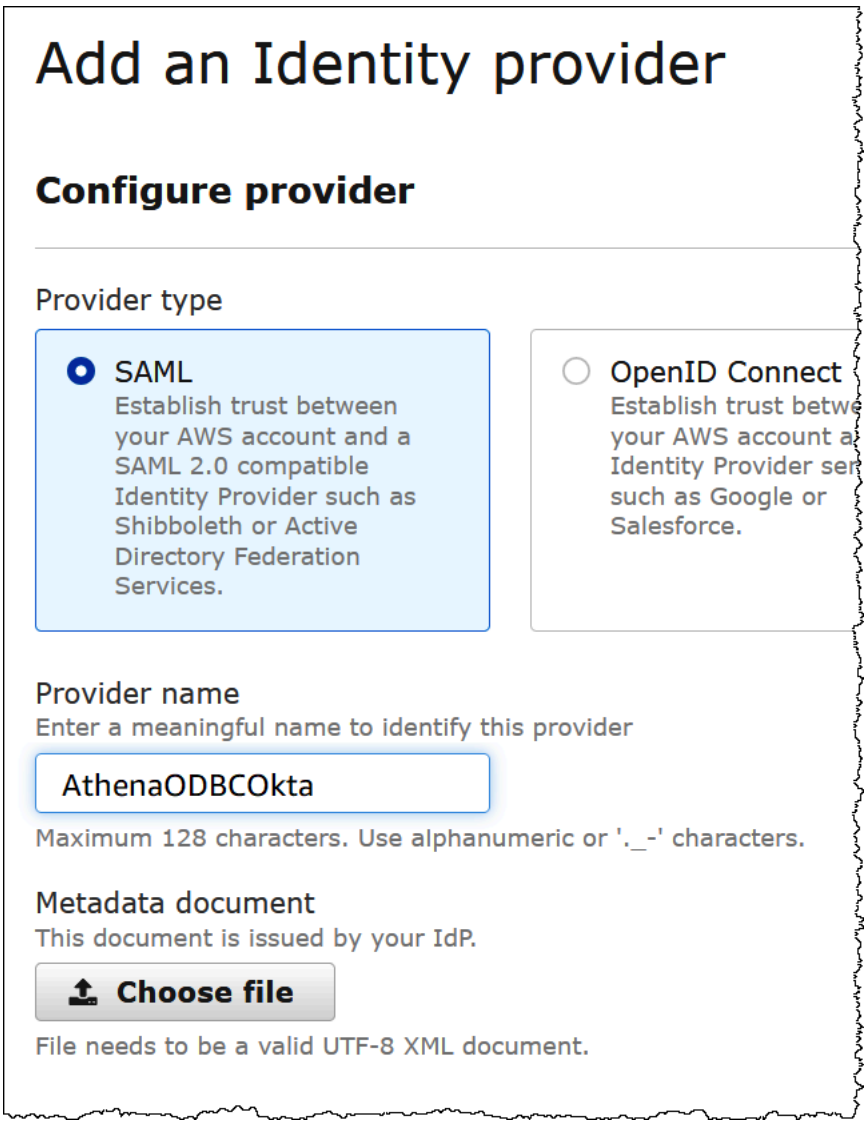
Pour créer un fournisseur d'identité et un rôle SAML dans AWS

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/IAM/.
2. Dans le panneau de navigation, sélectionnez Fournisseurs d'identité, puis Ajouter un fournisseur.



3. Sur la page Add an Identity provider (Ajouter un fournisseur d'identité), pour Configure provider (Configurer le fournisseur), saisissez les informations suivantes.

   - Pour Type de fournisseur, choisissez SAML.
   - Pour Provider name (Nom du fournisseur), saisissez un nom pour votre fournisseur (par exemple, **`AthenaODBCOkta`**).
   - Pour Metadata document (Document de métadonnées), utilisez l'option Choose file (Choisir un fichier) pour téléverser le fichier XML de métadonnées du fournisseur d'identité (IdP) que vous avez téléchargé.



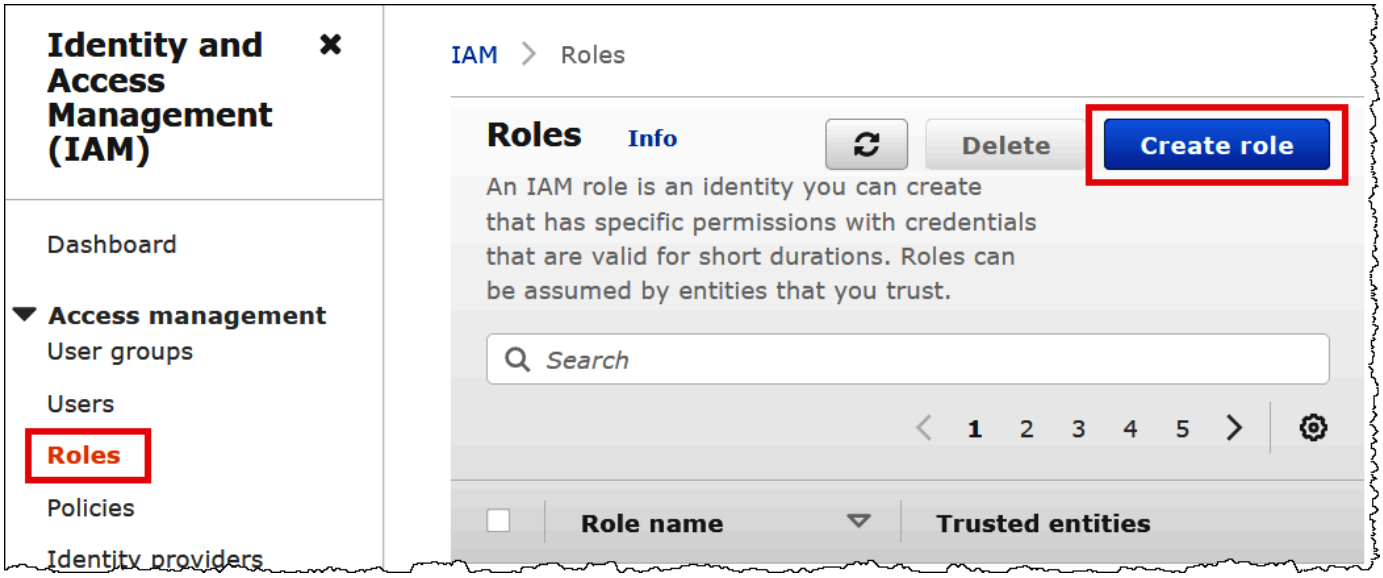
4. Choisissez Ajouter un fournisseur.

Création d'un rôle IAM pour l'accès à Athena et Amazon S3

Vous êtes maintenant prêt à créer un rôle IAM pour l'accès à Athena et Amazon S3. Vous allez attribuer ce rôle à votre utilisateur. De cette façon, vous pourrez fournir à l'utilisateur un accès à Athena par authentification unique.

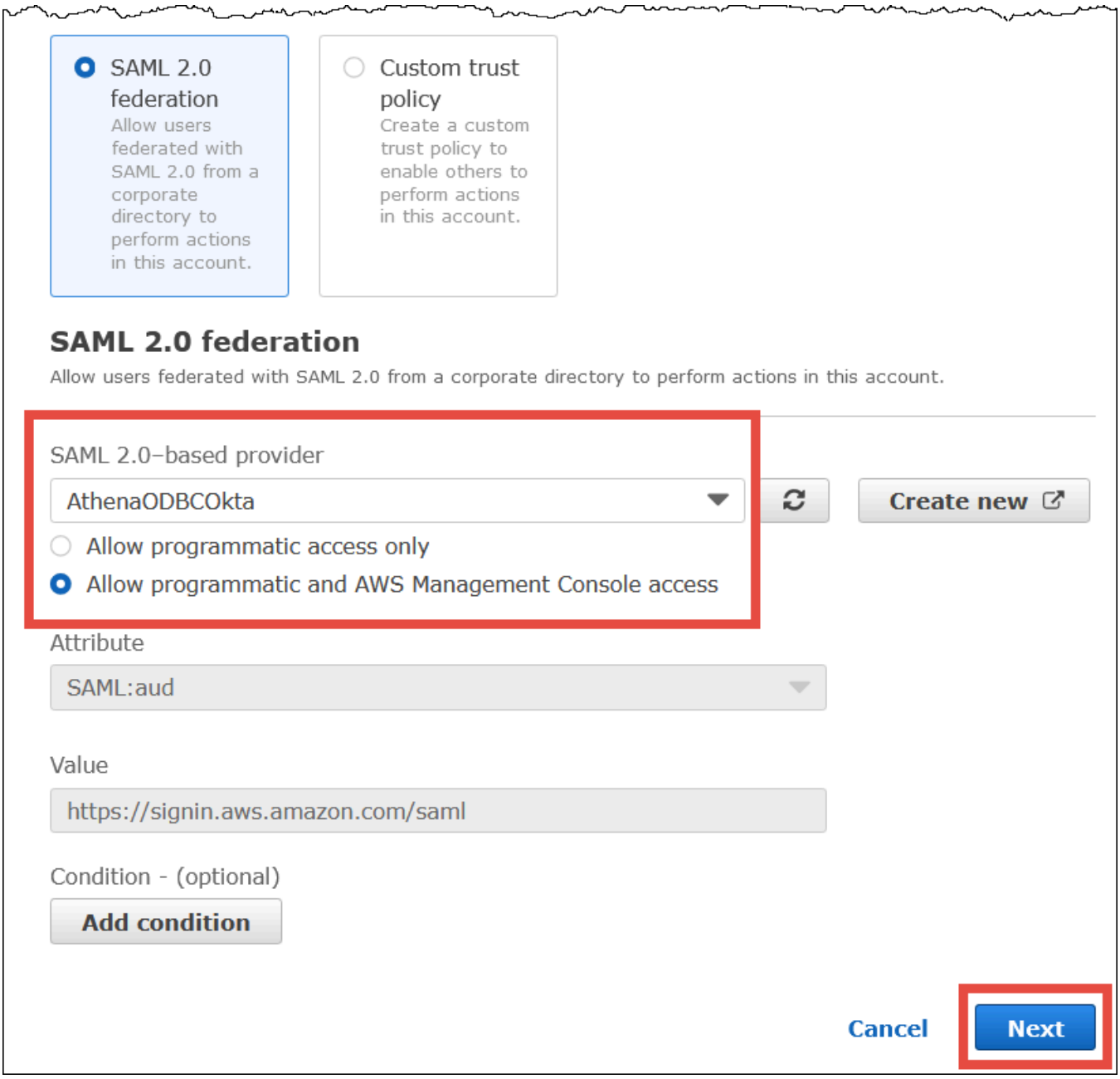
Pour créer un rôle IAM pour votre utilisateur

1. Dans le panneau de navigation de la console IAM, choisissez Roles (Rôles), puis Create role (Créer un rôle).

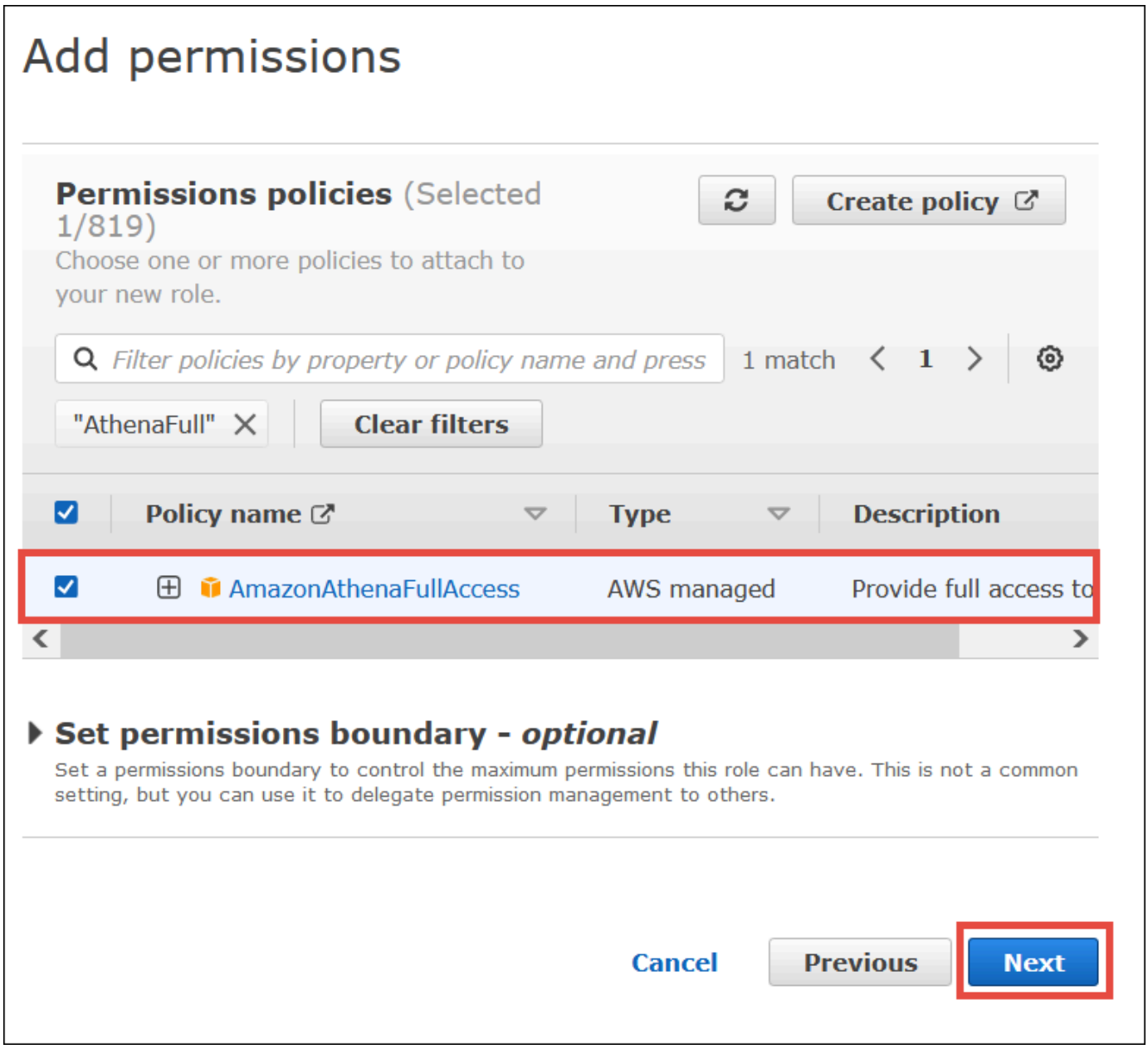


2. Sur la page Create role (Création d'un rôle), choisissez les options suivantes :

   - Pour Select type of trusted entity (Sélectionner le type d'entité de confiance), choisissez SAML 2.0 Federation.
   - Pour SAML 2.0-based provider (Fournisseur basé sur SAML 2.0), choisissez le fournisseur d'identité SAML que vous avez créé (par exemple, AthenaODBCOkta).
   - Sélectionnez Autoriser l'accès programmatique et à la AWS Management Console .



3. Choisissez Suivant.
4. Sur la page Add Permissions (Ajouter des autorisations), pour Filter policies (Politiques de filtrage), saisissez **`AthenaFull`**, puis appuyez sur ENTRÉE.
5. Sélectionnez la politique gérée `AmazonAthenaFullAccess`, puis cliquez sur Next (Suivant).



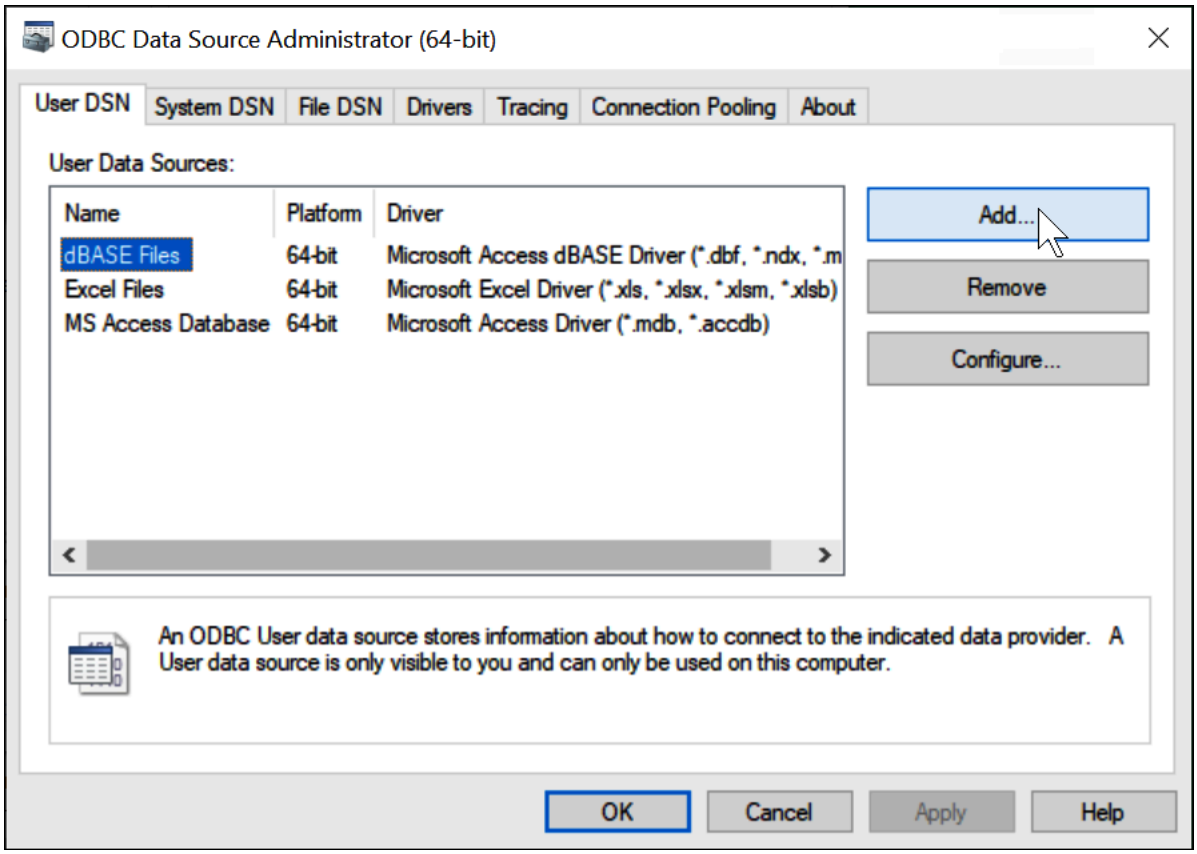
6. Sur la page Name, review, and create (Nom, révision et création), pour Role name (Nom du rôle), saisissez le nom du rôle (par exemple, **`Athena-ODBC-OktaRole`**), puis choisissez Create role (Créer un rôle).

Configuration de la connexion ODBC Okta à Athena

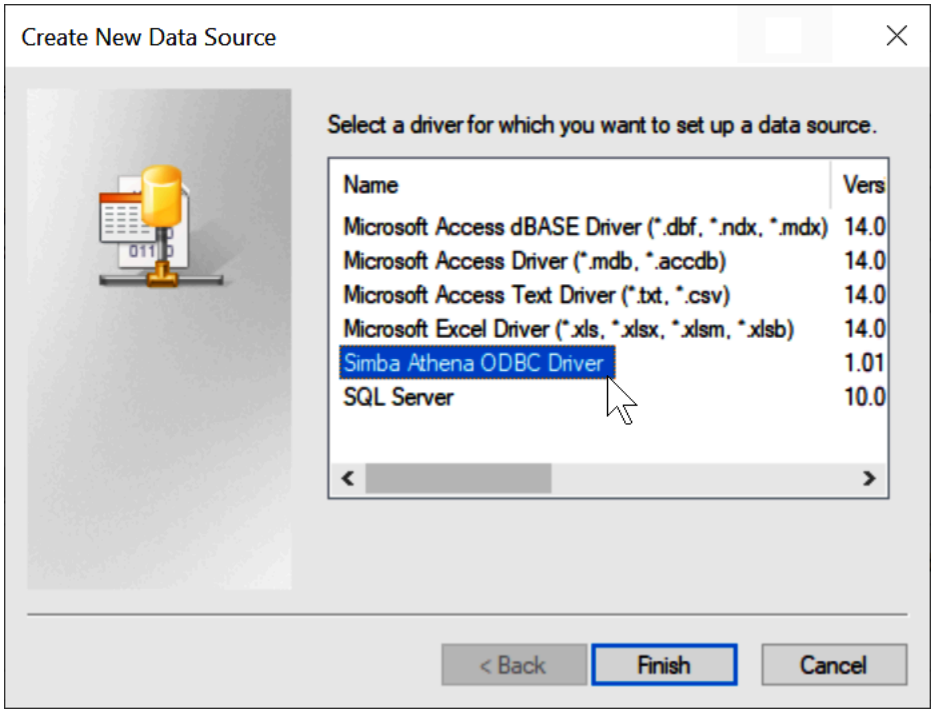
Vous êtes maintenant prêt à configurer la connexion ODBC Okta à Athena en utilisant le programme ODBC Data Sources sous Windows.

Pour configurer votre connexion ODBC Okta à Athena

1. Dans Windows, lancez le programme Sources de données ODBC.
2. Dans le programme Administrateur de source de données ODBC, choisissez Ajouter.

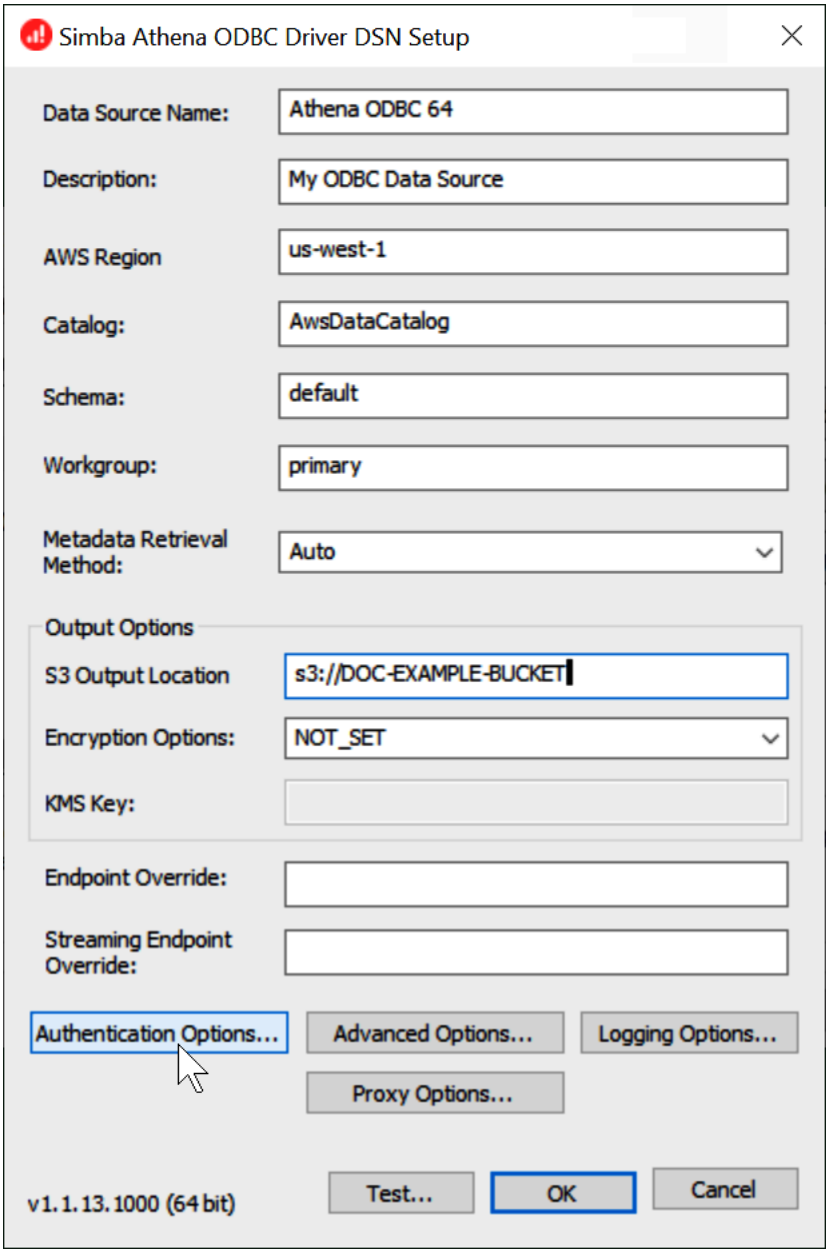


3. Choisissez Simba Athena ODBC Driver (Pilote ODBC Simba Athena), puis cliquez sur Finish (Terminer).



4. Dans la boite de dialogue Simba Athena ODBC Driver DSN Setup (Configuration DSN du pilote ODBC Simba Athena), saisissez les valeurs décrites.

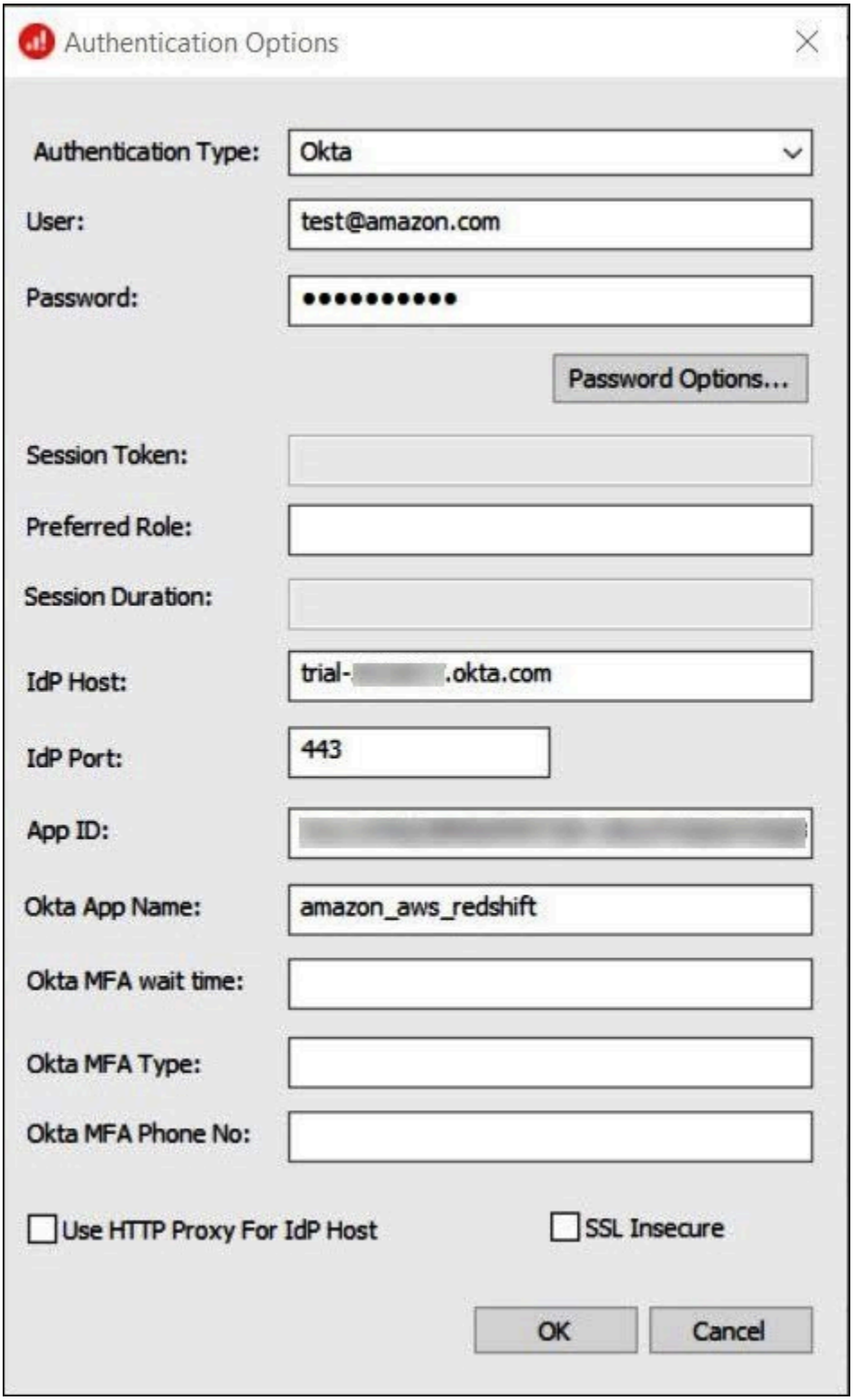
   - Pour Data Source Name (Nom de la source de données), saisissez un nom pour votre source de données (par exemple, **`Athena ODBC 64`**).
   - Pour Description, saisissez la description de votre source de données.
   - Pour Région AWS, entrez le Région AWS que vous utilisez (par exemple,**`us-west-1`**).
   - Pour Emplacement de sortie S3, saisissez le chemin Amazon S3 où vous souhaitez stocker votre sortie.



5. Choisissez Options d'authentification.
6. Dans la boite de dialogue Authentication Options (Options d'authentification), choisissez ou saisissez les valeurs suivantes.

   - Pour Authentication Type (Type d'authentification), choisissez Okta.
   - Pour Utilisateur, entrez votre nom d'utilisateur Okta.
   - Pour Mot de passe, entrez votre mot de passe Okta.
   - Pour IdP Host (Hôte IdP), saisissez la valeur que vous avez enregistrée précédemment (par exemple, **`trial-1234567.okta.com`**).

- Pour IdP Port (Port IdP), saisissez **`443`**.
- Pour App ID (ID d'application), saisissez la valeur que vous avez enregistrée précédemment (les deux derniers segments de votre lien intégré Okta).
- Pour Okta App Name (Nom de l'application Okta), saisissez **`amazon_aws_redshift`**.



7. Choisissez OK.
8. Choisissez Test (Tester) pour tester la connexion ou OK pour terminer.

Configuration de l'authentification unique à l'aide d'ODBC, SAML 2.0 et du fournisseur d'identité Okta

Pour vous connecter à des sources de données, vous pouvez utiliser Amazon Athena avec des fournisseurs d'identité (IdPs) tels PingOne qu'Okta OneLogin, etc. À partir du pilote ODBC Athena version 1.1.13 et du pilote Athena JDBC version 2.0.25, un plugin SAML de navigateur est inclus et peut être configuré pour fonctionner avec n'importe quel fournisseur SAML 2.0. Cette rubrique explique comment configurer le pilote ODBC Amazon Athena et le plugin SAML basé sur navigateur afin d'ajouter une fonctionnalité d'authentification unique (SSO) à l'aide du fournisseur d'identité d'Okta

Prérequis

Pour suivre les étapes de ce tutoriel, vous devez avoir :

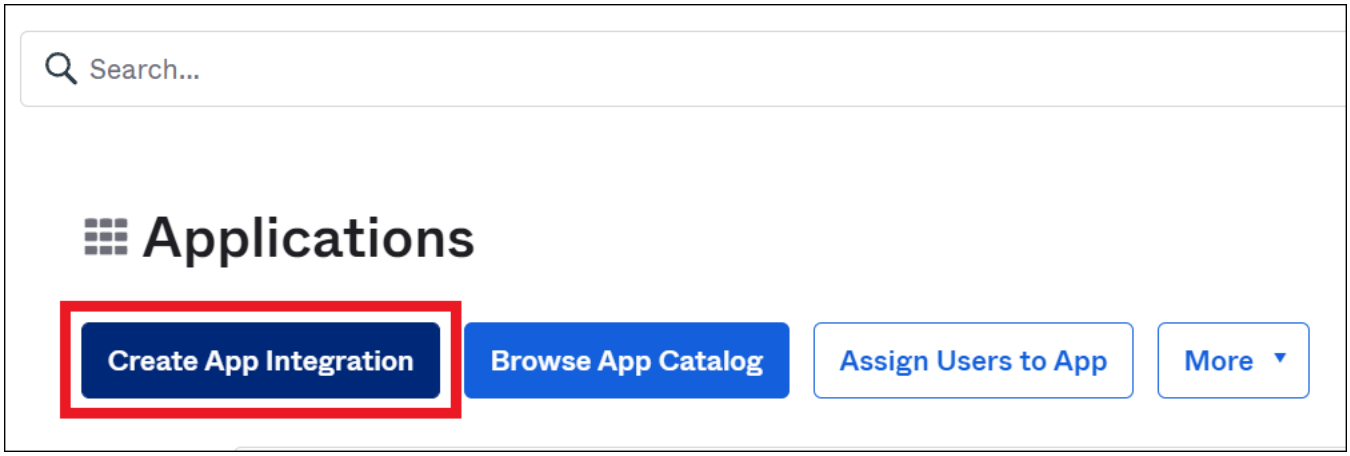
- Pilote ODBC Athena version 1.1.13 ou ultérieure. Les versions 1.1.13 et ultérieures incluent la prise en charge SAML du navigateur. Pour les liens de téléchargement, consultez la section Connexion à Amazon Athena avec ODBC.
- Un rôle IAM que vous souhaitez utiliser avec SAML. Pour plus d'informations, consultez la section Création d'un rôle pour la fédération SAML 2.0 du Guide de l'utilisateur IAM.
- Un compte Okta. Pour de plus amples informations, visitez okta.com.

Création d'une intégration d'appli dans Okta

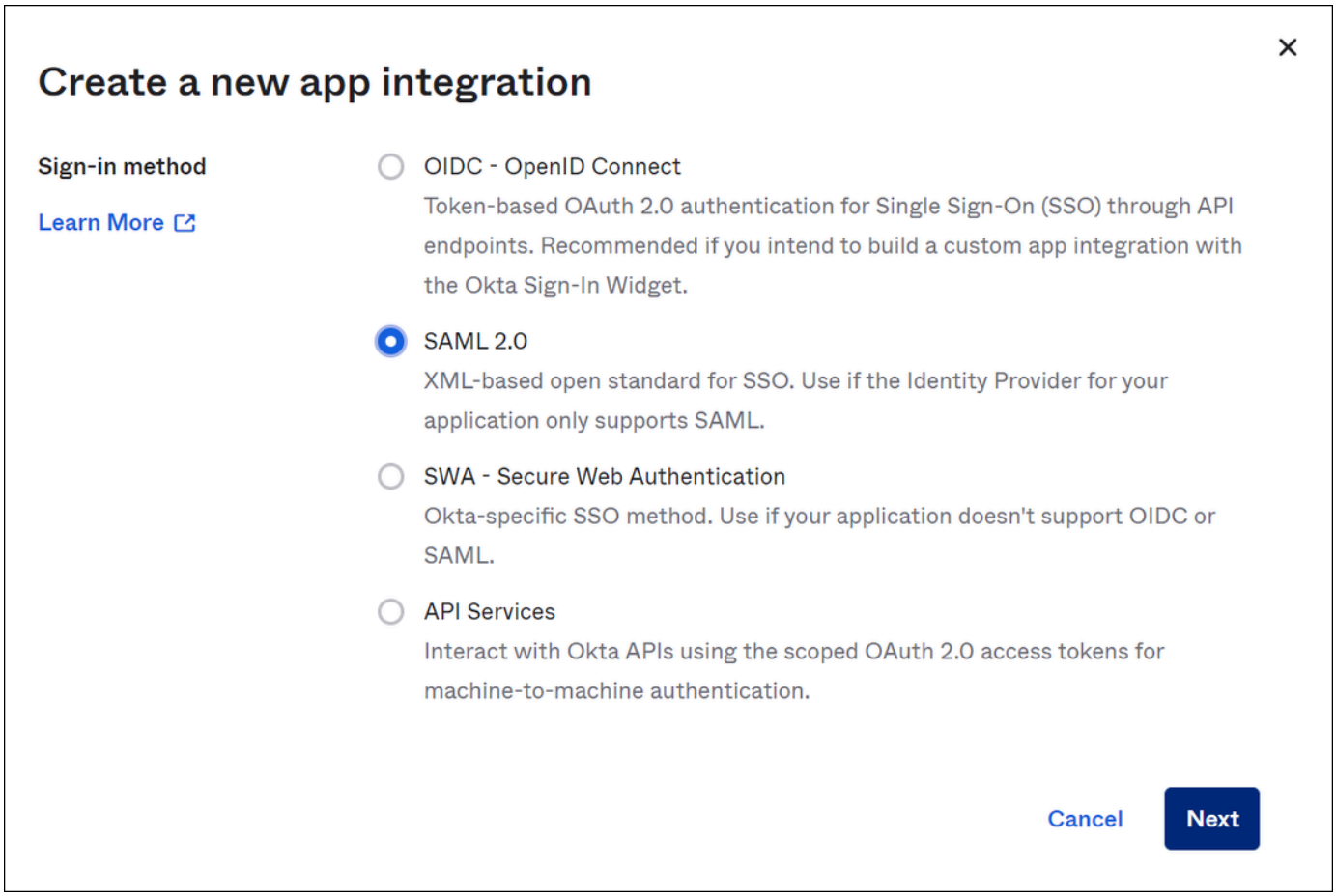
Tout d'abord, utilisez le tableau de bord Okta pour créer et configurer une appli SAML 2.0 pour l'authentification unique à Athena.

Pour utiliser le tableau de bord Okta afin de configurer l'authentification unique pour Athena

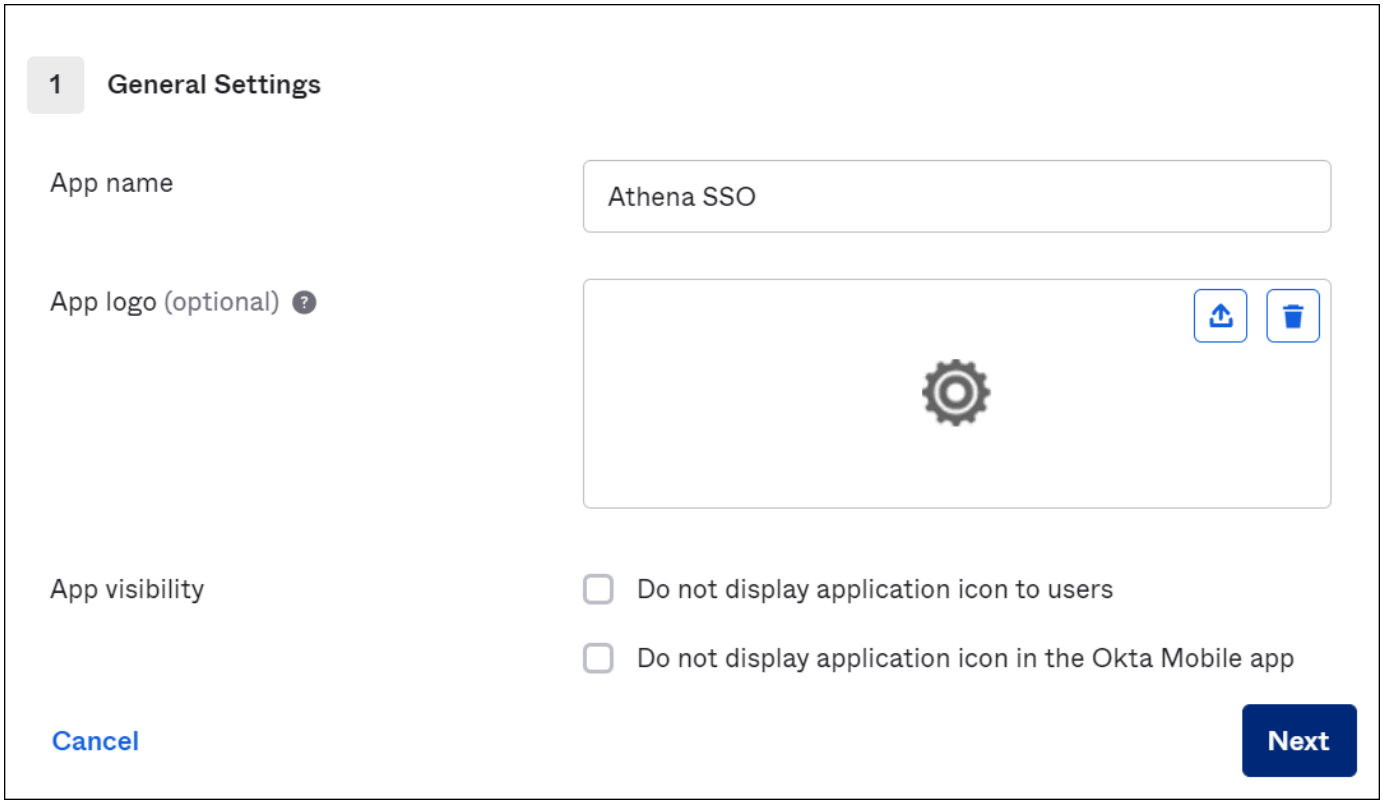
1. Connectez-vous à la page d'administration d'Okta à l'adresse `okta.com`.
2. Dans le panneau de navigation, choisissez Applications, Applications.
3. Sur la page Applications, choisissez Create App Integration (Créer une intégration d'appli).



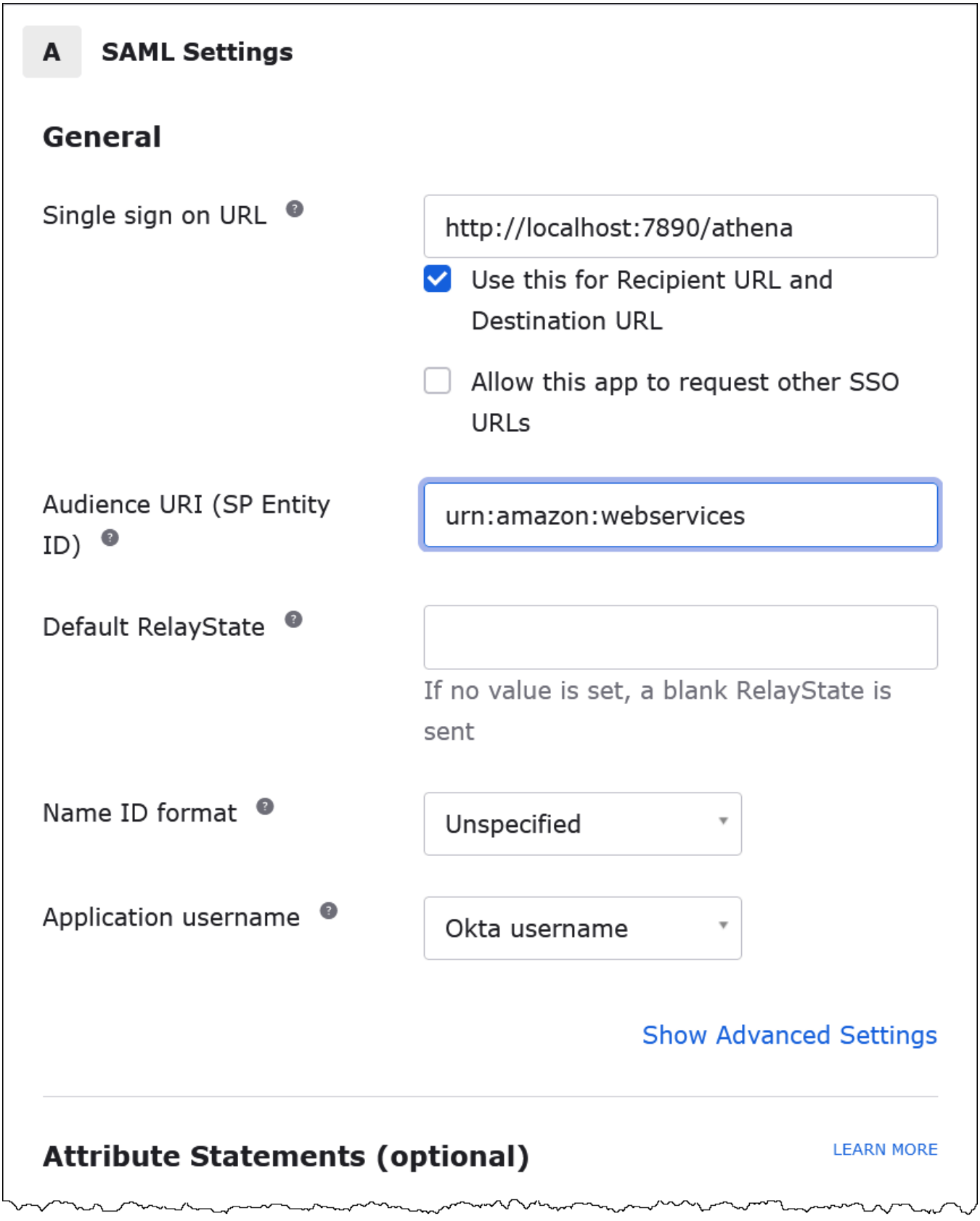
4. Dans la boîte de dialogue Create a new app integration (Créer une intégration d'appli), pour Sign-in method (Méthode de connexion), sélectionnez SAML 2.0, puis choisissez Next (Suivant).



5. Sur la page Create SAML Integration (Créer une intégration SAML), dans la section General Settings (Paramètres généraux), saisissez un nom pour l'application. Ce didacticiel utilise le nom SSO Athena.



6. Choisissez Suivant.
7. Sur la page Configurer SAML, dans la sectionSAML Settings (Paramètres SAML), saisissez les valeurs suivantes :
   - Pour Single sign on URL (URL d'authentification unique), saisissez **`http://localhost:7890/athena`**
   - Pour Audience URI (URI du public), saisissez **`urn:amazon:webservices`**


A
SAML Settings
General
Single sign on URL
http://localhost:7890/athena
Use this for Recipient URL and Destination URL
Allow this app to request other SSO URLs
Audience URI (SP Entity ID)
urn:amazon:webservices
Default RelayState
If no value is set, a blank RelayState is sent
Name ID format
Unspecified
Application username
Okta username
Show Advanced Settings
Attribute Statements (optional)
LEARN MORE

8. Pour Attribute Statements (optional) [Déclarations d'attributs (facultatif)], saisissez les deux paires nom-valeur suivantes : Il s'agit d'attributs de mappage obligatoires.

   - Pour Name (Nom), saisissez l'URL suivante :

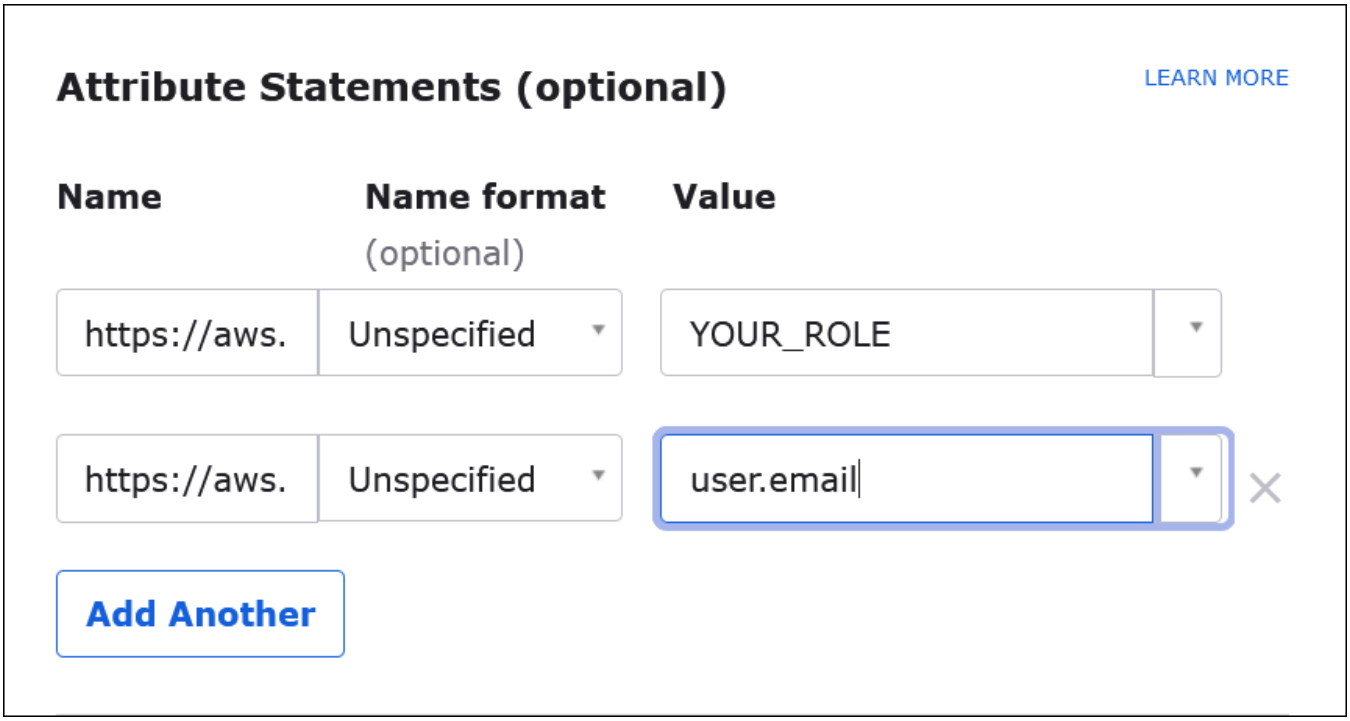
     `https://aws.amazon.com/SAML/Attributes/Role`

     Pour Value (Valeur), saisissez le nom de votre rôle IAM. Pour plus d'informations sur le format des rôles IAM, consultez la section Configuration des assertions SAML pour la réponse d'authentification dans le Guide de l'utilisateur IAM.

   - Pour Nom, saisissez l'URL suivante :

     `https://aws.amazon.com/SAML/Attributes/RoleSessionName`

     Pour le champ Value (Valeur), saisissez **`user.email`**.



9. Cliquez sur Suivant, puis sur Terminer.

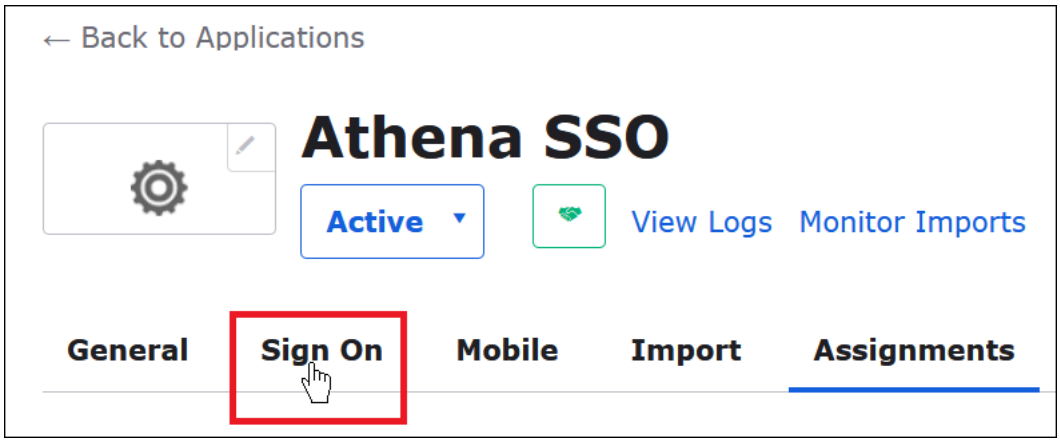
   Lorsque Okta crée l'application, elle crée également votre URL de connexion, que vous récupérerez ensuite.

Obtention de l'URL de connexion depuis le tableau de bord Okta

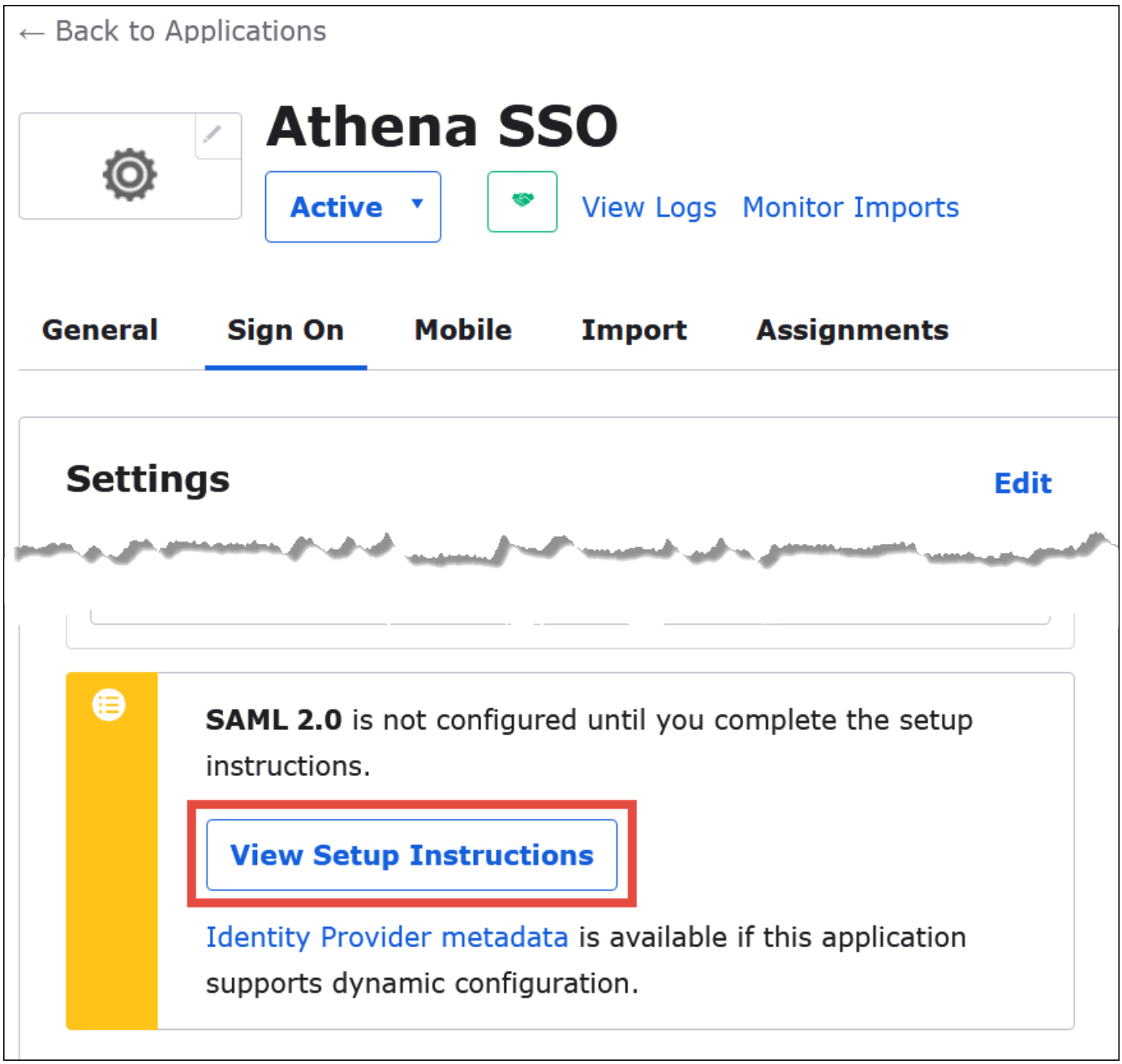
Maintenant que votre application a été créée, vous pouvez obtenir son URL de connexion et d'autres métadonnées à partir du tableau de bord Okta.

Pour obtenir l'URL de connexion à partir du tableau de bord Okta

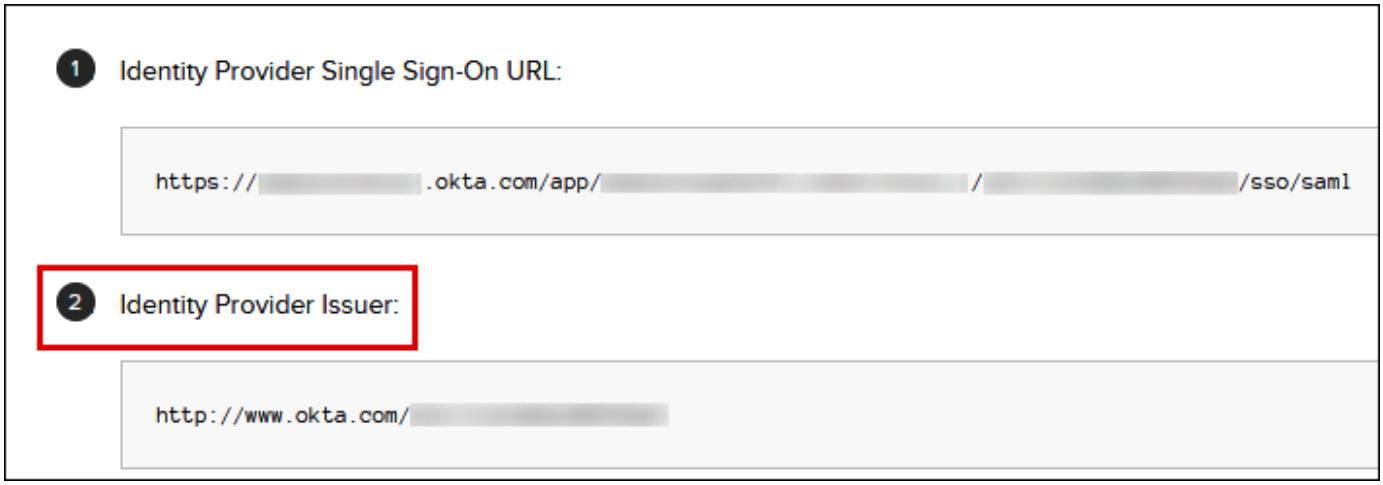
1. Dans le panneau de navigation Okta, choisissez Applications, Applications.
2. Choisissez l'application pour laquelle vous voulez trouver l'URL de connexion (par exemple, AthenaSSO).
3. Sur la page de votre application, choisissez Sign On (Connexion).



4. Choisissez Afficher les instructions de configuration.



5. Sur la page How to Configure SAML 2.0 for Athena SSO (Comment configurer SAML 2.0 pour Athena SSO), recherchez l'URL pour Identity Provider Issuer (Émetteur du fournisseur d'identité). Certains endroits du tableau de bord Okta font référence à cette URL comme étant l'ID du SAML issuer ID (Émetteur SAML).



6. Copiez ou stockez la valeur pour Identity Provider Single Sign-On URL. (URL d'authentification unique du fournisseur d'identité)

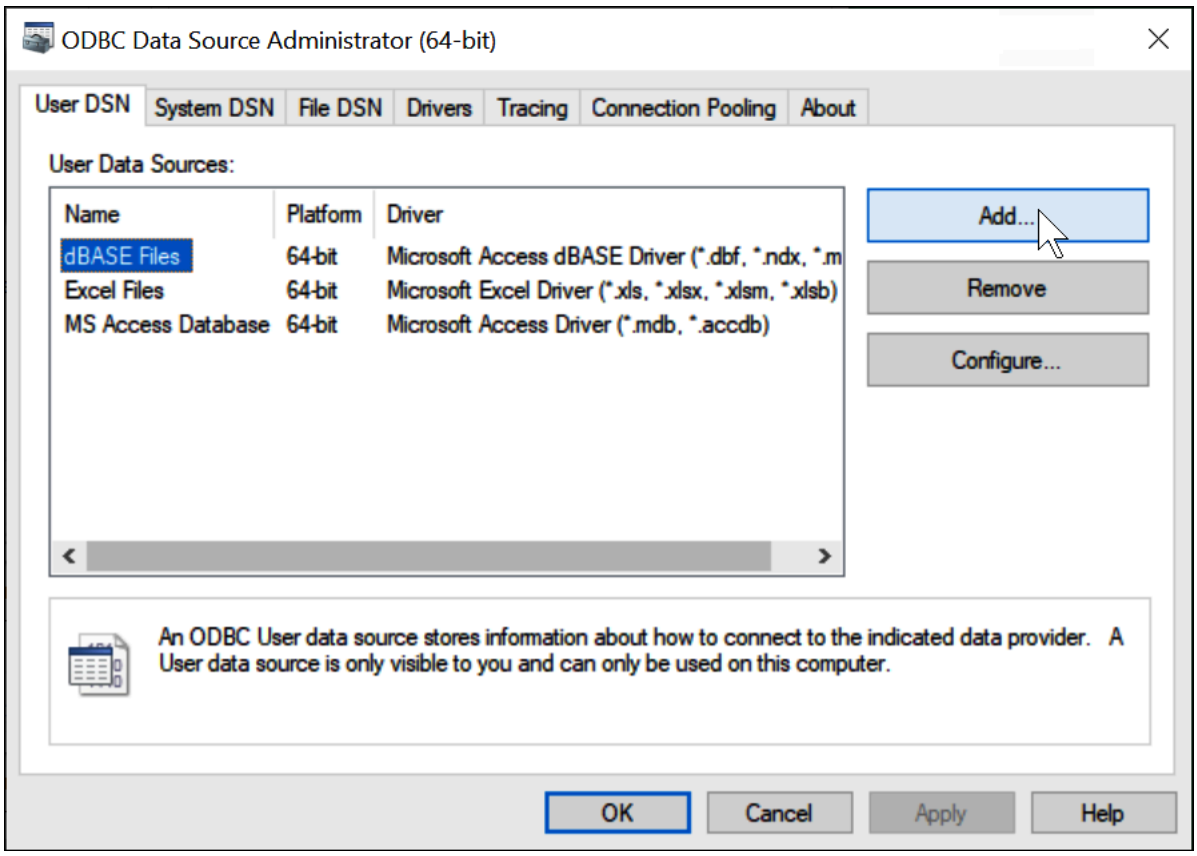
   Dans la section suivante, lorsque vous configurerez la connexion ODBC, vous fournirez cette valeur comme paramètre de connexion de l'URL de connexion pour le plugin SAML du navigateur.

Configuration de la connexion ODBC SAML du navigateur à Athena

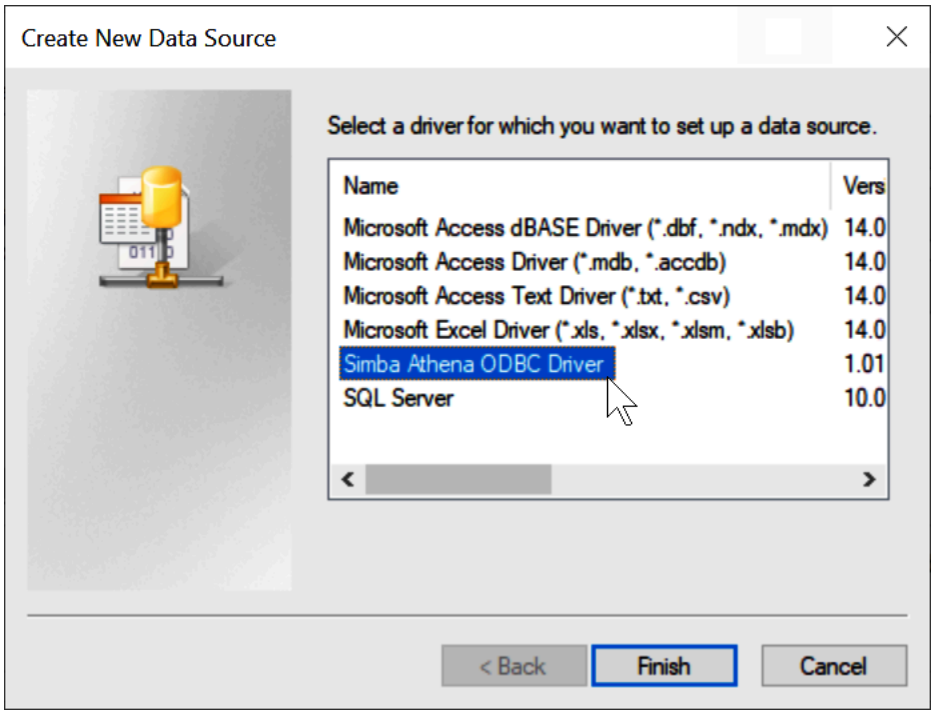
Vous êtes maintenant prêt à configurer la connexion SAML du navigateur à Athena à l'aide du programme Sources de données ODBC sous Windows.

Pour configurer la connexion ODBC SAML du navigateur à Athena

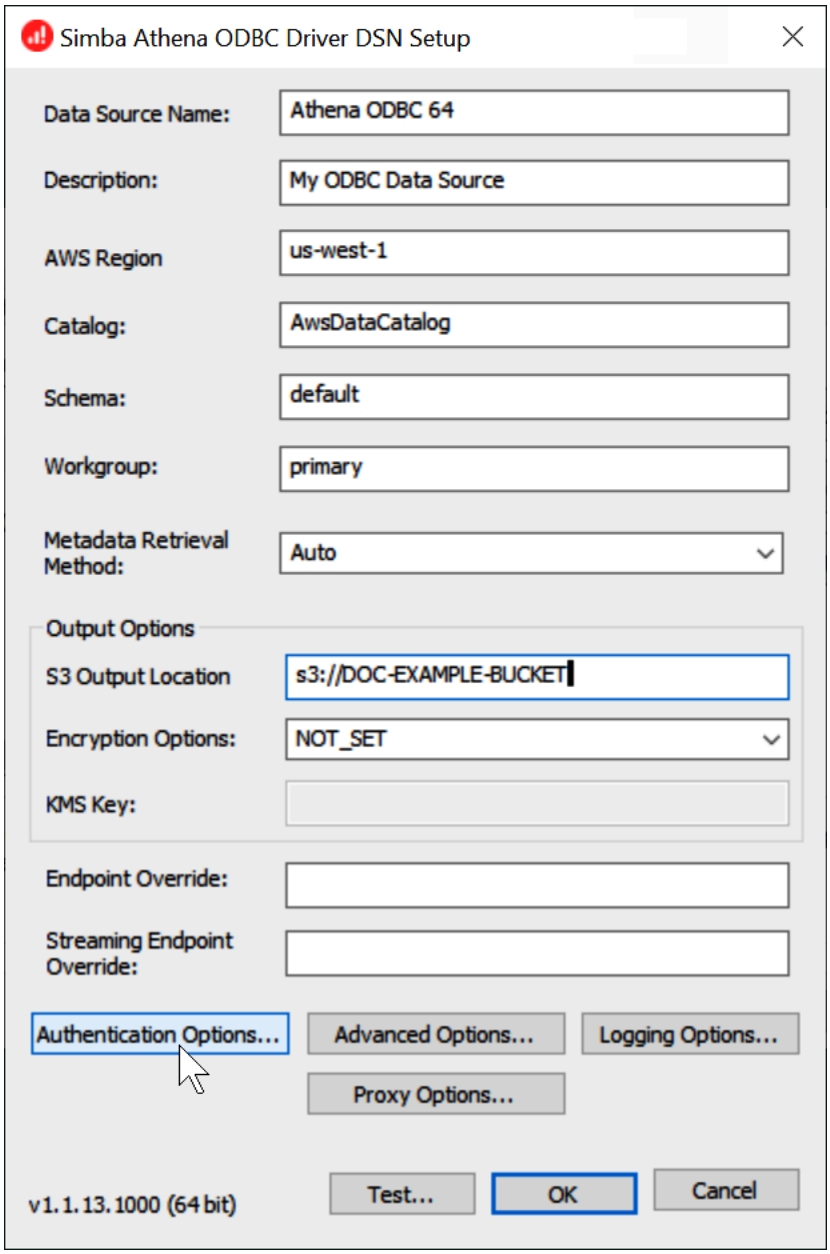
1. Dans Windows, lancez le programme Sources de données ODBC.
2. Dans le programme Administrateur de source de données ODBC, choisissez Ajouter.



3. Choisissez Simba Athena ODBC Driver (Pilote ODBC Simba Athena), puis cliquez sur Finish (Terminer).



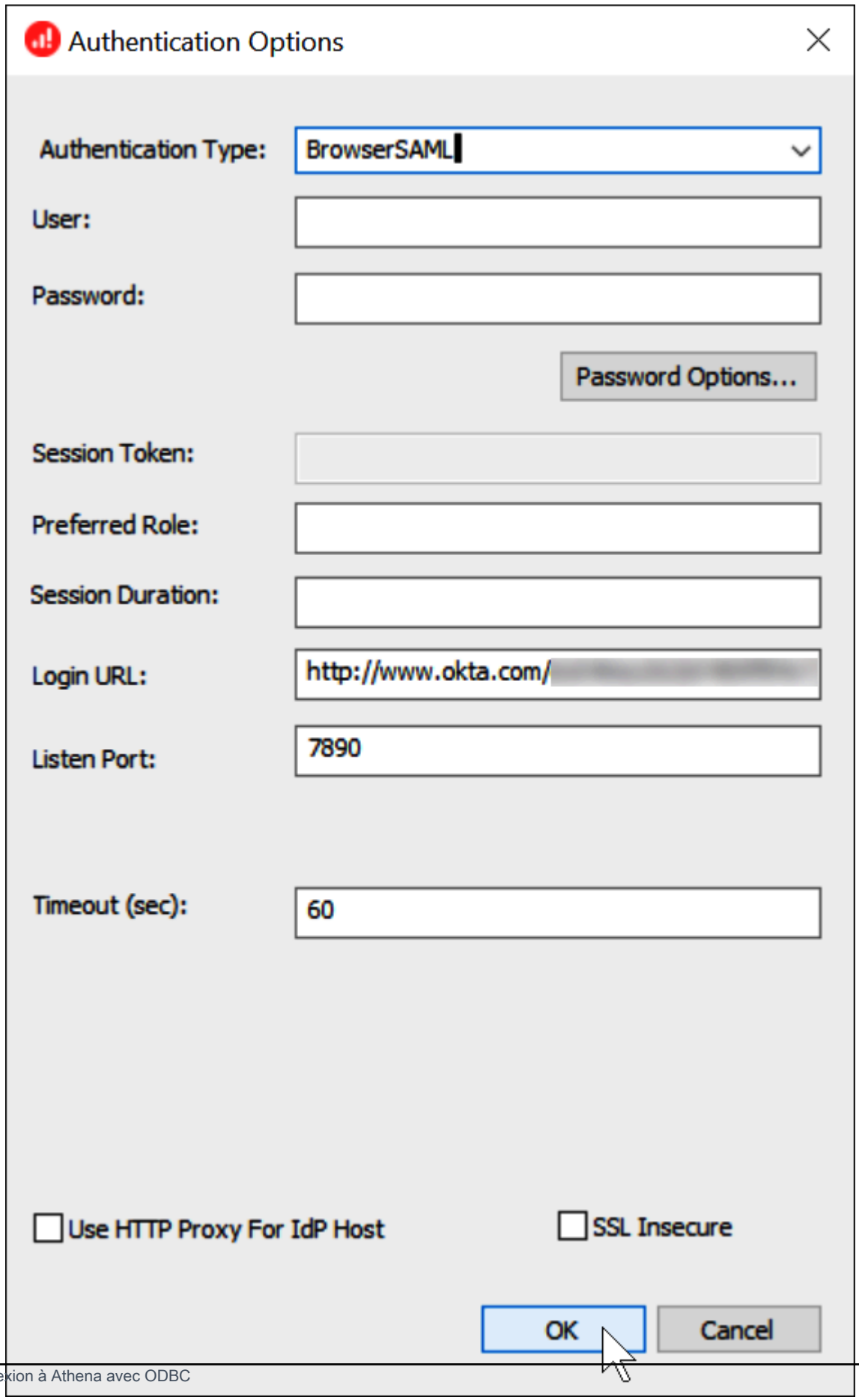
4. Dans la boite de dialogue Simba Athena ODBC Driver DSN Setup (Configuration DSN du pilote ODBC Simba Athena), saisissez les valeurs décrites.



- Pour Data Source Name (Nom de la source de données), saisissez un nom pour votre source de données (par exemple, Athena ODBC 64).
- Pour Description, saisissez la description de votre source de données.
- Pour Région AWS, entrez le Région AWS que vous utilisez (par exemple,**`us-west-1`**).
- Pour S3 Output Location (Emplacement de sortie S3), saisissez le chemin Amazon S3 où vous souhaitez stocker votre sortie.

5. Choisissez Authentication Options (Options d'authentification).

6. Dans la boite de dialogue Authentication Options (Options d'authentification), choisissez ou saisissez les valeurs suivantes.


Authentication Options
Authentication Type:
BrowserSAML
User:
Password:
Password Options...
Session Token:
Preferred Role:
Session Duration:
Login URL:
http://www.okta.com/
Listen Port:
7890
Timeout (sec):
60
Use HTTP Proxy For IdP Host
SSL Insecure
OK
Cancel

- Pour Authentication Type (Type d'authentification), choisissez BrowserSAML (Navigateur SAML).
- Pour Login URL (URL de connexion), saisissez l'Identity Provider Single Sign-On URL (URL d'authentification unique du fournisseur d'identité) que vous avez obtenu à partir du tableau de bord Okta.
- Pour Listen Port (Port d'écoute), saisissez 7890.
- Pour Timeout (sec) [Délai d'attente (s)], saisissez une valeur de délai d'expiration de connexion en secondes.

7. Choisissez OK pour fermer les Authentication Options (Options d'authentification).
8. Choisissez Test (Tester) pour tester la connexion, ou OK pour terminer.

Utilisation du connecteur Amazon Athena Power BI

Sur les systèmes d'exploitation Windows, vous pouvez utiliser le connecteur Microsoft Power BI pour Amazon Athena pour analyser les données d'Amazon Athena dans Microsoft Power BI Desktop. Pour plus d'informations sur Power BI, consultez la rubrique Microsoft power BI. Après avoir publié du contenu sur le service Power BI, vous pouvez utiliser la version de juillet 2021 ou ultérieure de Power BI gateway pour maintenir le contenu à jour grâce à des rafraîchissements à la demande ou programmés.

Prérequis

Avant de commencer, assurez-vous que votre environnement répond aux exigences suivantes. Le pilote ODBC Amazon Athena est requis.

- Compte AWS
- Autorisations d'utiliser Athena
- Pilote ODBC Amazon Athena
- Power BI desktop

Fonctionnalités prises en charge

- Import – Les tables et les colonnes sélectionnées sont importées dans Power BI Desktop à des fins d'interrogation.

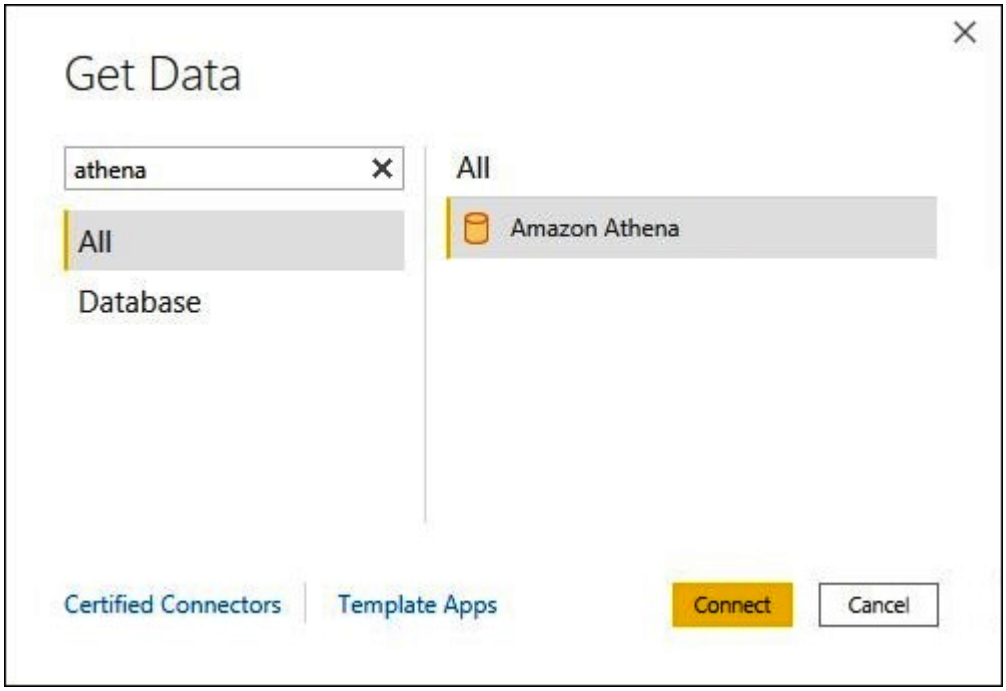
- DirectQuery— Aucune donnée n'est importée ou copiée dans Power BI Desktop. Power BI Desktop interroge directement la source de données sous-jacente.
- Passerelle Power BI : passerelle de données sur site intégrée à votre Compte AWS ordinateur qui fonctionne comme un pont entre le service Microsoft Power BI et Athena. La passerelle est nécessaire pour voir vos données sur le service Power BI de Microsoft.

Connexion à Amazon Athena

Pour connecter Power BI Desktop à vos données Amazon Athena, procédez comme suit.

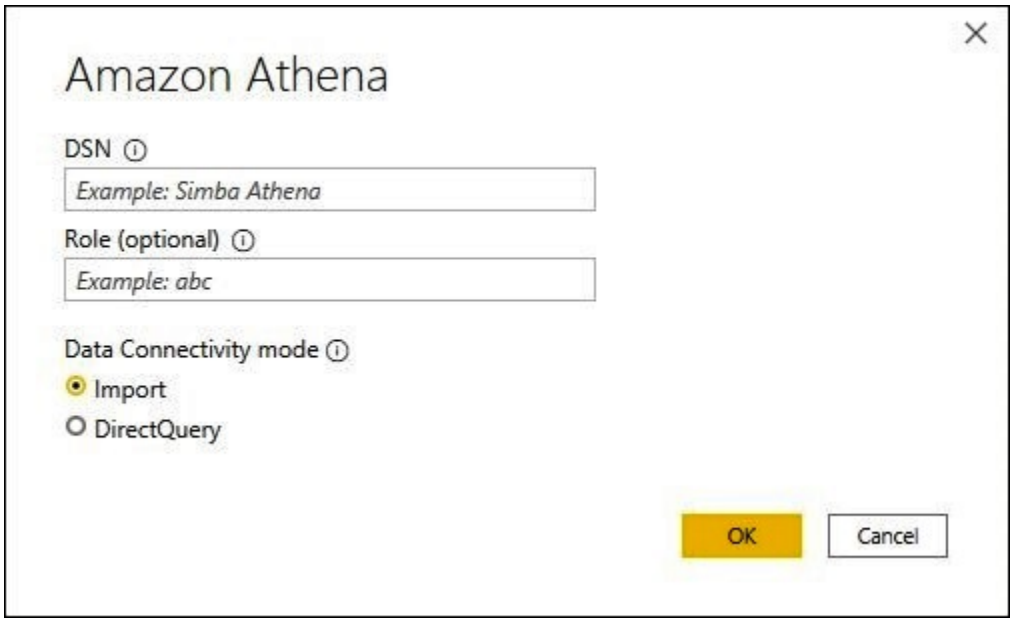
Connexion aux données Athena à partir de power BI desktop

1. Lancez Power BI Desktop.
2. Effectuez l’une des actions suivantes :
   - Choisissez File (Fichier), Get Data (Obtenir des données)
   - Dans le ruban Home (Accueil), choisissez Get Data (Obtenir des données).
3. Dans la zone de recherche, saisissez Athena.
4. Sélectionnez Amazon Athena, puis choisissez Connect (Connexion).

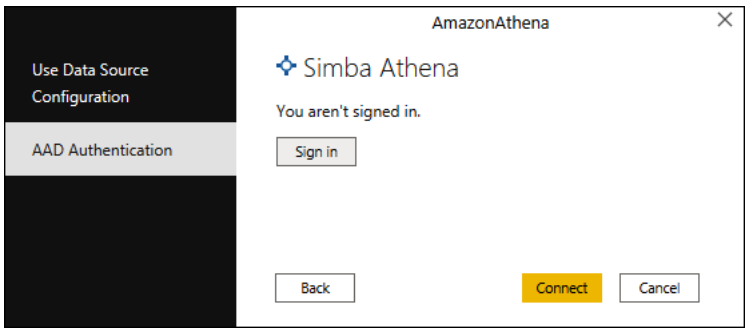


5. Sur la page de connexion Amazon Athena, saisissez les informations suivantes.

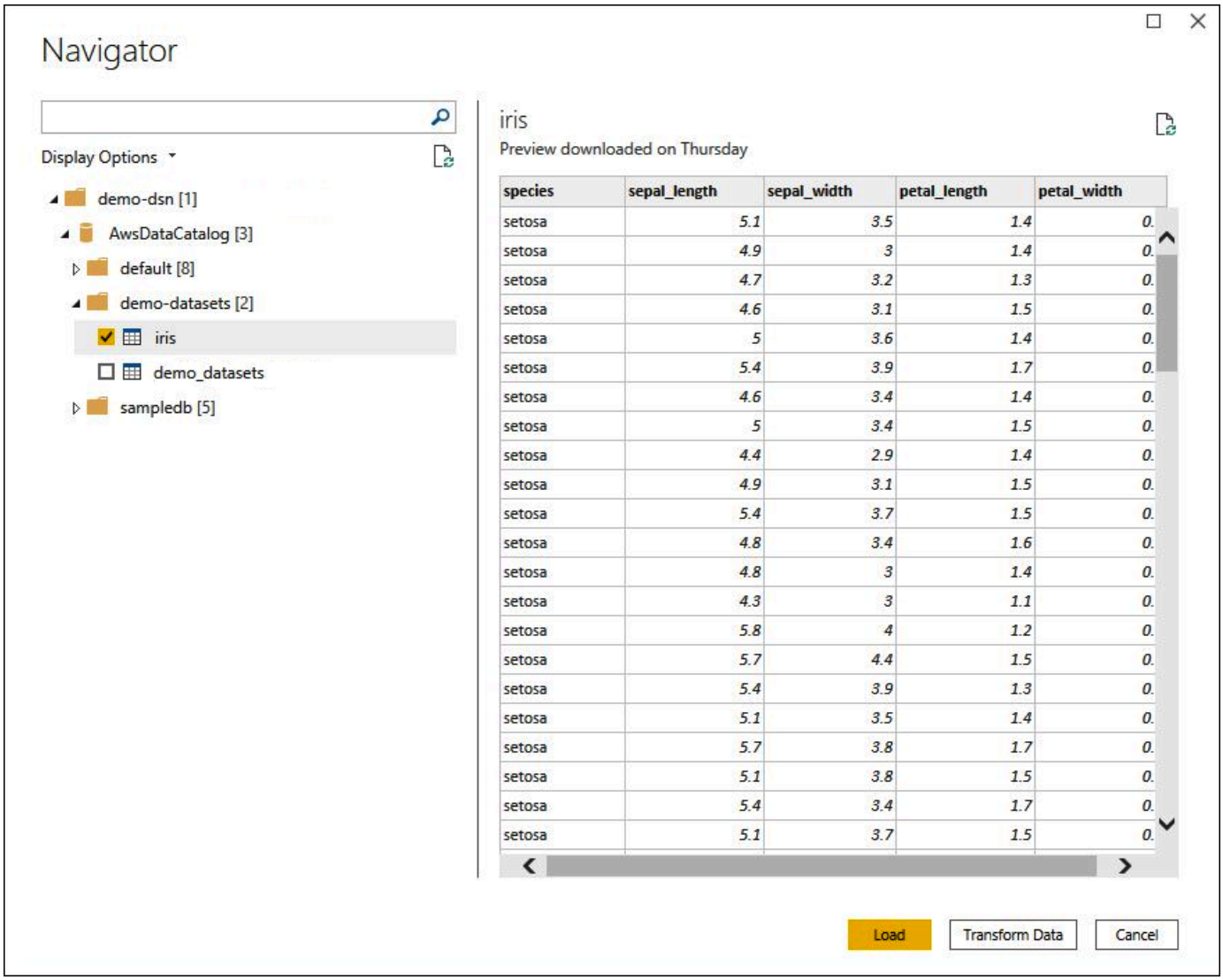
- Pour DSN, saisissez le nom du DSN ODBC que vous souhaitez utiliser. Pour obtenir des instructions sur la configuration de votre DSN, consultez la documentation du pilote ODBC.
- Pour le mode Connectivité des données, choisissez un mode qui convient à votre cas d'utilisation, en suivant ces instructions générales :
  - Pour les jeux de données plus petits, choisissez Import (Importation). En mode Importation, Power BI travaille avec Athena pour importer le contenu de l'ensemble du jeu de données afin de l'utiliser dans vos visualisations.
  - Pour les ensembles de données plus volumineux, choisissez DirectQuery. En DirectQuery mode, aucune donnée n'est téléchargée sur votre poste de travail. Pendant que vous créez ou interagissez avec une visualisation, Microsoft Power BI travaille avec Athena pour interroger dynamiquement la source de données sous-jacente, de sorte que vous visualisez toujours les données actuelles. Pour plus d'informations DirectQuery, consultez la section Utilisation DirectQuery dans Power BI Desktop dans la documentation Microsoft.



6. Choisissez OK.
7. À l'invite de configuration de l'authentification de la source de données, choisissez soit Use Data Source Configuration (Utilisation de la configuration de la source de données), soit AAD Authentication (Authentification AAD), puis cliquez sur Connect (Connexion).

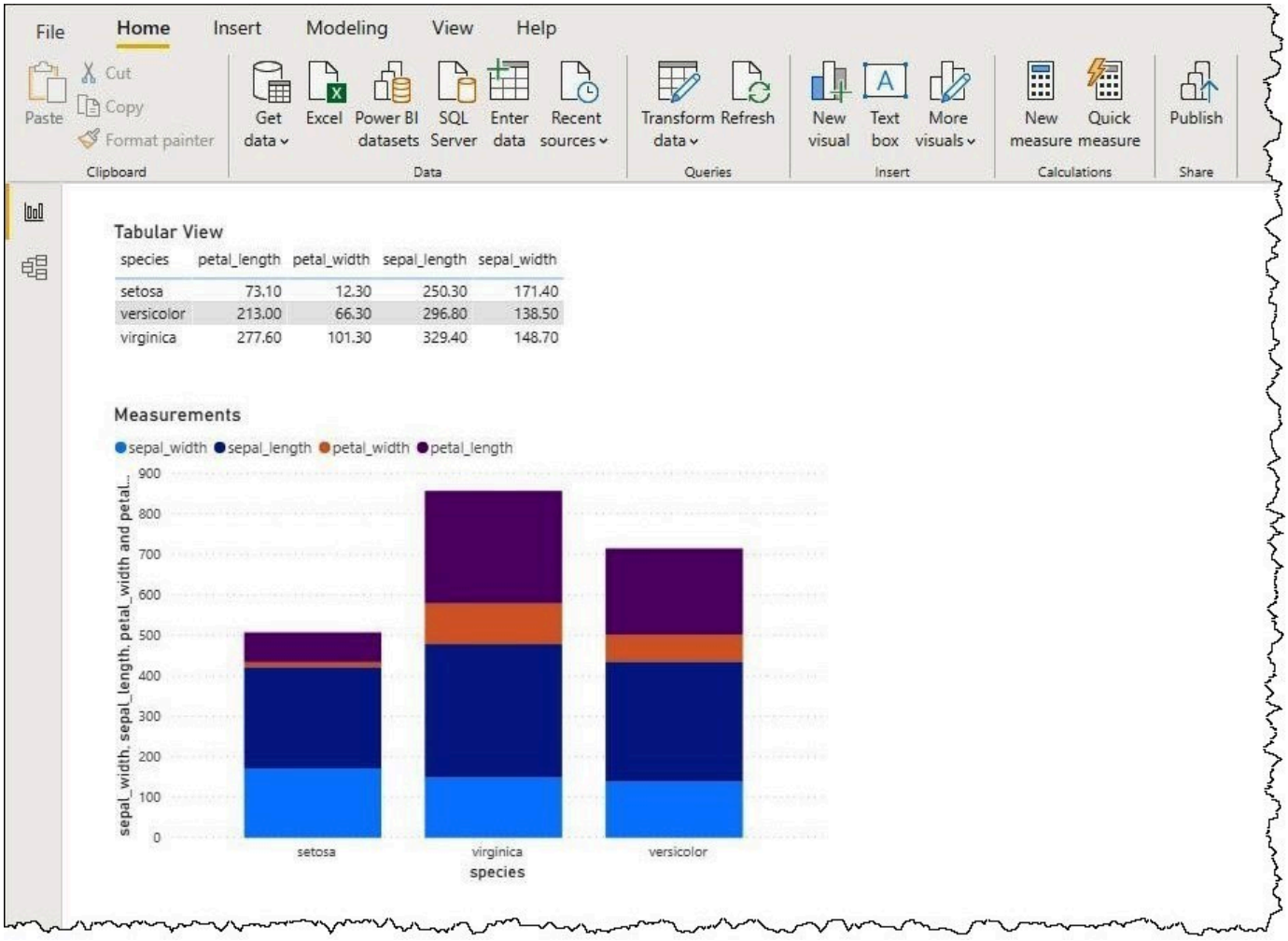


Votre catalogue de données, vos bases de données et vos tables apparaissent dans la boîte de dialogue Navigator (Navigateur).



8. Dans le volet Display Options (Options d'affichage), cochez la case correspondant au jeu de données que vous souhaitez utiliser.

9. Si vous voulez transformer le jeu de données avant de l'importer, allez au bas de la boîte de dialogue et choisissez Transform Data (Transformer les données). L'éditeur de requêtes Power Query s'ouvre alors pour vous permettre de filtrer et d'affiner le jeu de données que vous souhaitez utiliser.
10. Choisissez Load (Charger). Une fois le chargement terminé, vous pouvez créer des visualisations comme celle de l'image suivante. Si vous avez sélectionné DirectQueryle mode d'importation, Power BI envoie une requête à Athena pour la visualisation que vous avez demandée.



Configuration d'une passerelle sur site

Vous pouvez publier des tableaux de bord et des jeux de données sur le service Power BI afin que d'autres utilisateurs puissent interagir avec eux via des applications Web, mobiles et intégrées. Pour voir vos données dans le service Microsoft Power BI, vous devez installer la passerelle de données

Microsoft Power BI sur site dans votre Compte AWS. La passerelle fonctionne comme un pont entre le service Microsoft Power BI et Athena.

Téléchargement, installation et test d'une passerelle de données sur site

1. Visitez la page Téléchargement de la passerelle Microsoft power BI et choisissez le mode personnel ou le mode standard. Le mode personnel est utile pour tester localement le connecteur Athena. Le mode standard est approprié dans un contexte de production multi-utilisateurs.
2. Pour installer une passerelle sur site (en mode personnel ou standard), consultez la rubrique Installation d'une passerelle de données sur site dans la documentation Microsoft.
3. Pour tester la passerelle, suivez les étapes de la rubrique Utilisation de connecteurs de données personnalisés avec la passerelle de données sur site de la documentation Microsoft.

Pour plus d'informations sur les passerelles de données sur site, consultez les ressources Microsoft suivantes.

- Qu'est-ce qu'une passerelle de données sur site ?
- Conseils pour le déploiement d'une passerelle de données pour power BI

Pour un exemple de configuration de Power BI Gateway pour une utilisation avec Athena, consultez l'article du blog AWS Big Data intitulé Création rapide de tableaux de bord sur Microsoft Power BI à l'aide d'Amazon Athena.

Création de bases de données et de tables

Amazon Athena prend en charge un sous-ensemble d'instructions DDL (langage de définition de données) et de fonctions et opérateurs SQL ANSI pour définir et interroger des tables externes dans lesquelles résident les données Amazon Simple Storage Service.

Lorsque vous créez une table de base de données dans Athena, vous décrivez le schéma et l'emplacement des données, ce qui rend les données de table prêtes pour l'interrogation au moment de la lecture.

Pour améliorer les performances des requêtes et réduire les coûts, nous vous recommandons de partitionner vos données et d'utiliser des formats de stockage en colonnes open source pour le stockage dans Simple Storage Service (Amazon S3), par exemple Apache parquet ou ORC.

Rubriques

Création de bases de données dans Athena

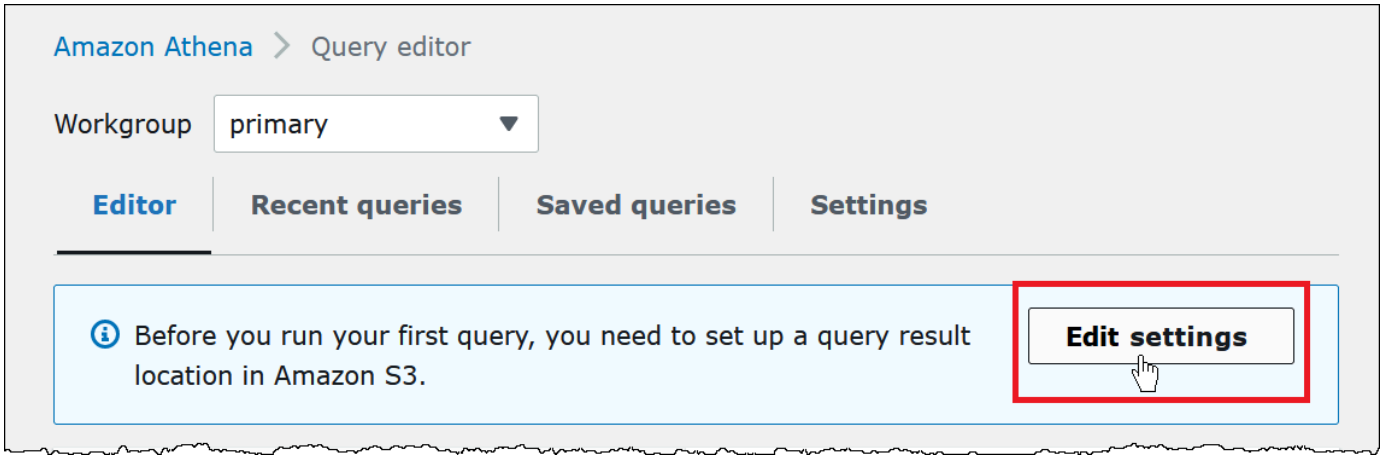
Dans Athena, une base de données est un regroupement logique des tables que vous y créez.

Prérequis

Si aucun emplacement de sortie de requête n'est déjà configuré dans Amazon S3, effectuez les étapes préalables suivantes pour ce faire.

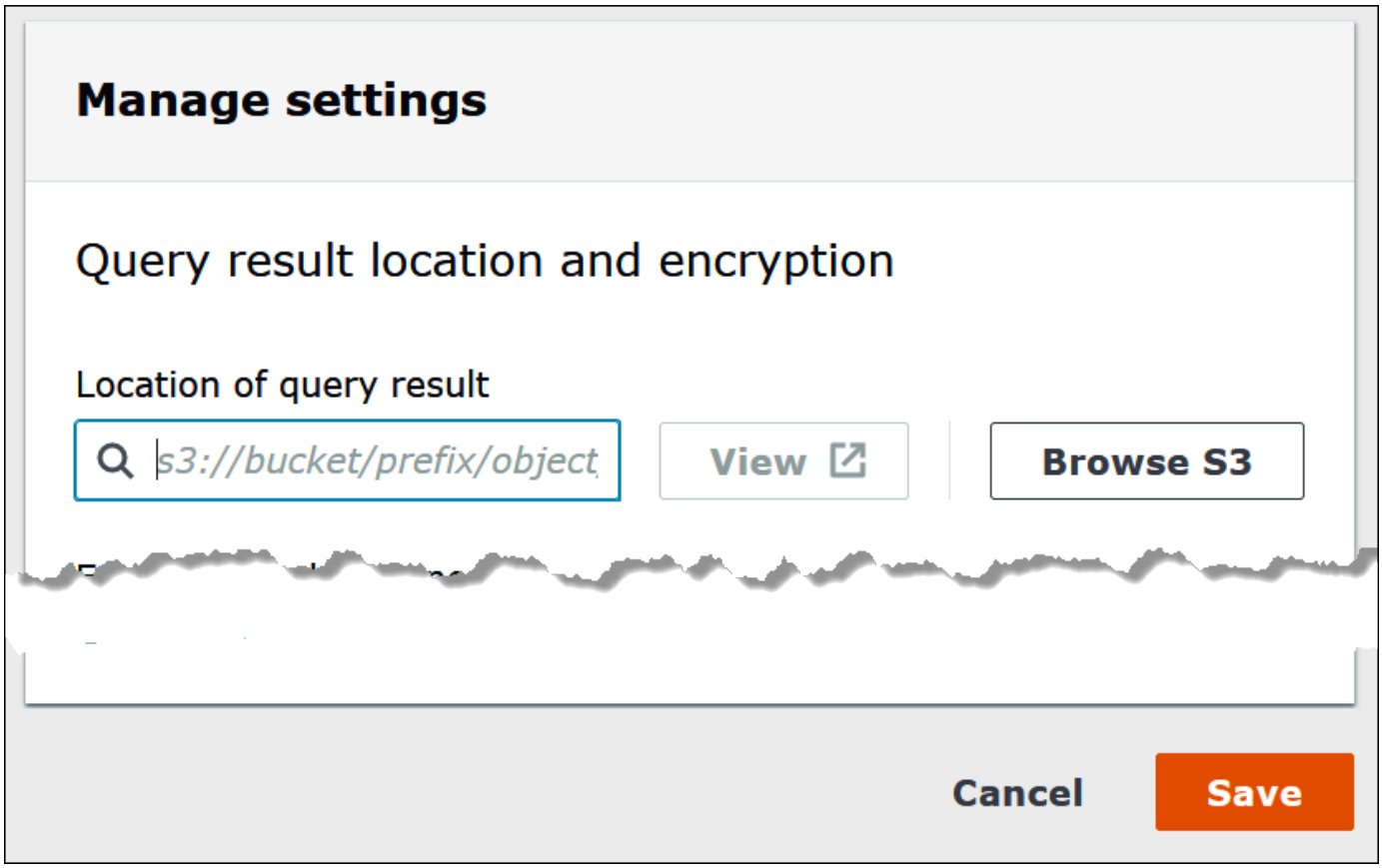
Pour créer un emplacement de sortie de requête

1. En utilisant la même Région AWS et le même compte que vous utilisez pour Athena, suivez les étapes (par exemple, en utilisant la console Amazon S3) pour créer un compartiment dans Amazon S3 pour les résultats de votre requête Athena. Vous allez configurer ce compartiment pour qu'il soit l'emplacement de sortie de votre requête.
2. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
3. S'il s'agit de votre première visite sur la console Athena dans votre Région AWS actuelle, choisissez Découvrir l'éditeur de requêtes pour ouvrir l'éditeur de requêtes. Sinon, Athena s'ouvre dans l'éditeur de requêtes.
4. Choisissez Modifier les paramètres pour configurer l'emplacement des résultats de la requête dans Amazon S3.

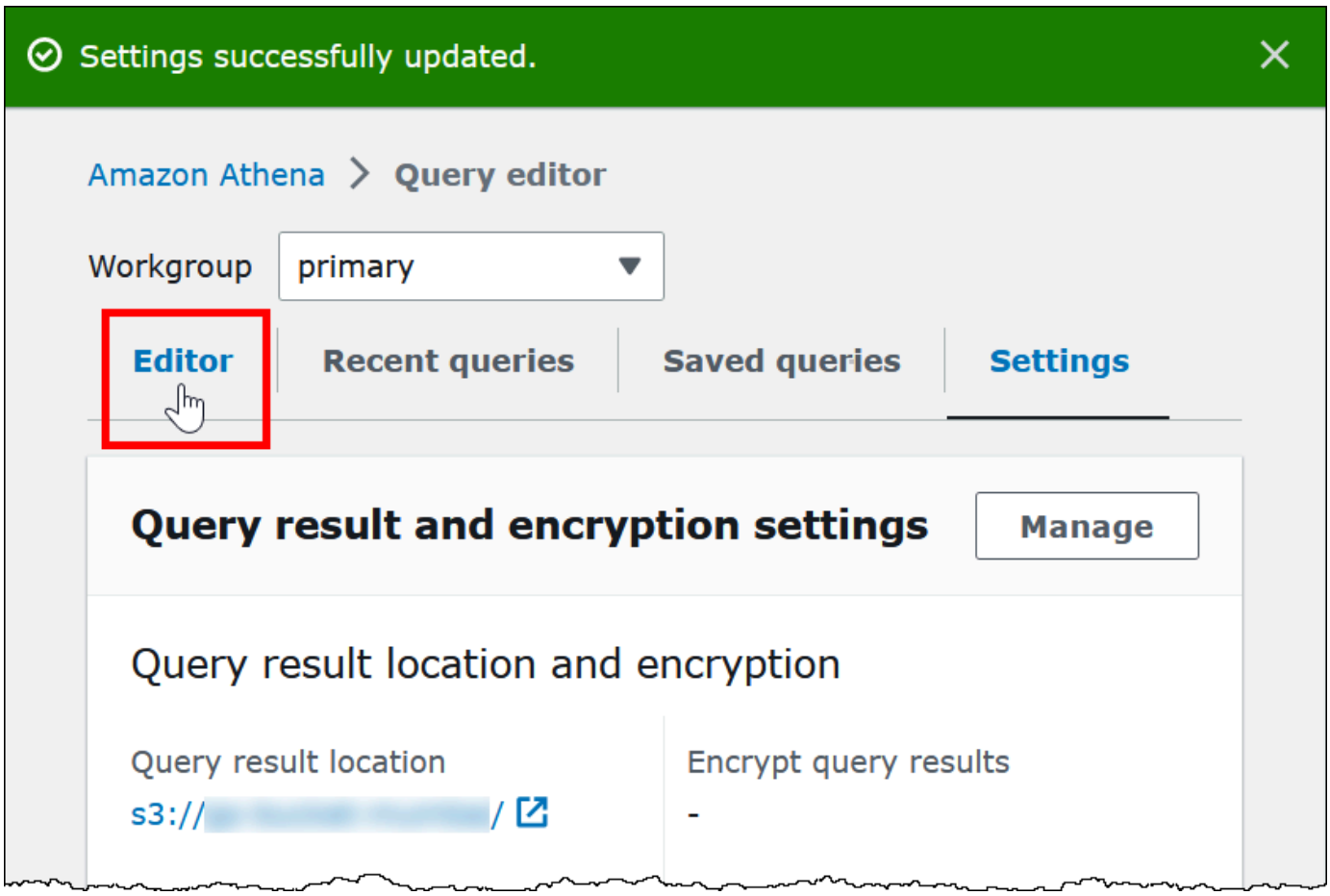


5. Pour Gestion des paramètres, procédez de l'une des manières suivantes :

   - Dans la zone de texte Location of query result (Emplacement des résultats de la requête), saisissez le chemin d'accès au compartiment que vous avez créé dans Simple Storage Service (Amazon S3) pour les résultats de votre requête. Préfixez le chemin avec `s3://`.
   - Choisissez Parcourir S3, choisissez le compartiment Simple Storage Service (Amazon S3) que vous avez créé pour votre région actuelle, puis sélectionnez Choisir.



6. Choisissez Enregistrer.
7. Choisissez Editor (Éditeur) pour passer à l'éditeur de requête.



Création d'une base de données

Une fois que vous avez configuré l'emplacement des résultats d'une requête, la création d'une base de données dans l'éditeur de requêtes de la console Athena est simple.

Création d'une base de données à l'aide de l'éditeur de requêtes Athena

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans l'onglet Éditeur de l'éditeur de requêtes, entrez la commande Data Definition Language (DDL) Hive `CREATE DATABASE myDataBase`. Remplacez *myDatabase* par le nom que vous souhaitez utiliser. Pour connaître les restrictions sur les noms de bases de données, consultez Noms des tables, des bases de données et des colonnes.
3. Choisissez Run (Exécuter) ou appuyez sur **Ctrl+ENTER**.

4. Pour faire de votre base de données la base de données actuelle, sélectionnez-la dans le menu Database (Base de données) à gauche de l'éditeur de requêtes.

Pour plus d'informations sur le contrôle des autorisations des bases de données Athena, consultez Accès détaillé aux bases de données et aux tables du AWS Glue Data Catalog.

Création de tables dans Athena

Vous pouvez exécuter des déclarations DDL dans la console Athena à l'aide d'un pilote JDBC ou ODBC ou à l'aide du formulaire Create table (Créer une table) d'Athena.

Lorsque vous créez un schéma de table dans Athena, Athena le stocke dans un catalogue de données et l'utilise lorsque vous exécutez des requêtes.

Athena utilise une approche connue sous le nom de schema-on-read, qui signifie qu'un schéma est projeté sur vos données au moment où vous exécutez une requête. Cela élimine la nécessité de charger ou transformer des données.

Athena ne modifie pas vos données dans Simple Storage Service (Amazon S3).

Athena utilise Apache Hive pour définir des tables et créer des bases de données, qui sont essentiellement des espaces de noms logiques de tables.

Lorsque vous créez une table de base de données dans Athena, vous décrivez simplement le schéma et l'emplacement où les données de table sont situées dans Simple Storage Service (Amazon S3) pour l'interrogation au moment de la lecture. Les notions de base de données et de table ont donc une signification légèrement différente par rapport aux systèmes de base de données relationnelle classiques, car les données ne sont pas stockées avec la définition de schéma pour la base de données et la table.

Lorsque vous effectuez une requête, vous interrogez la table à l'aide du code SQL standard et les données sont lues à ce moment. Vous trouverez des conseils pour créer des bases de données et des tables dans la documentation Apache Hive, mais les conseils suivants s'appliquent spécifiquement à Athena.

La longueur de chaîne de requête maximum est de 256 ko.

Hive prend en charge plusieurs formats de données grâce à l'utilisation des bibliothèques serializer-deserializer (). SerDe Vous pouvez également définir des schémas complexes à l'aide d'expressions

régulières. Pour obtenir la liste des SerDe bibliothèques prises en charge, consultezSerDe et formats de données pris en charge.

Considérations et restrictions

Voici quelques limitations et aspects importants à prendre en compte pour les tables dans Athena.

Exigences pour les tables dans Athena et les données dans Simple Storage Service (Amazon S3)

Lorsque vous créez une table, vous spécifiez un emplacement de compartiment Simple Storage Service (Amazon S3) pour les données sous-jacentes à l'aide de la clause `LOCATION`. Éléments à prendre en compte :

- Athena peut uniquement interroger la dernière version des données sur un compartiment Simple Storage Service (Amazon S3) versionné, et ne peut pas interroger les versions précédentes des données.
- Vous devez disposer des autorisations appropriées pour utiliser des données dans l'emplacement Simple Storage Service (Amazon S3). Pour plus d'informations, consultez Accès à Amazon S3 depuis Athena.
- Athena prend en charge l'interrogation des objets stockés avec plusieurs classes de stockage dans le même compartiment spécifié par la clause `LOCATION`. Par exemple, vous pouvez interroger des données dans les objets stockés dans différentes classes de stockage (Standard, Standard – Accès peu fréquent et Hiérarchisation intelligente) dans Simple Storage Service (Amazon S3).
- Athena prend en charge les compartiments de type Paiement par le demandeur. Pour savoir comment activer « Requester Pays » pour les compartiments contenant les données source que vous souhaitez interroger dans Athena, consultez Créer un groupe de travail.
- Athena ne prend pas en charge l'interrogation des données dans les classes de stockage S3 Glacier Flexible Retrieval ou S3 Glacier Deep Archive. Les objets dans les classes de stockage S3 Glacier Flexible Retrieval et S3 Glacier Deep Archive sont ignorés. Vous pouvez également utiliser la classe de stockage Simple Storage Service (Amazon S3) Glacier Instant Retrieval, qui peut être interrogée par Athena. Pour de plus amples informations, consultez classe de stockage Simple Storage Service (Amazon S3) Glacier instant retrieval.

  Pour plus d'informations sur les classes de stockage, reportez-vous aux rubriques Classes de stockage, Modification de la classe de stockage d'un objet dans Amazon S3, Transition vers la classe de stockage GLACIER (archivage d'objets) et Compartiments de type paiement par le demandeur du Guide de l'utilisateur Amazon Simple Storage Service.

- Si vous exécutez des requêtes sur les compartiments Simple Storage Service (Amazon S3) avec un grand nombre d'objets et que les données ne sont pas partitionnées, de telles requêtes peuvent affecter les limites de débit de demande Get dans Simple Storage Service (Amazon S3) et conduire à des exceptions Simple Storage Service (Amazon S3). Pour éviter des erreurs, partitionnez vos données. En outre, envisagez de régler vos taux de demande Simple Storage Service (Amazon S3). Pour plus d'informations, consultez Considérations en matière de débit de demandes et de performances.
- Si vous utilisez l'opération AWS Glue CreateTableAPI ou le AWS CloudFormation `AWS::Glue::Table`modèle pour créer une table à utiliser dans Athena sans spécifier la `TableType` propriété, puis si vous exécutez une requête DDL du type `SHOW CREATE TABLE` ou`MSCK REPAIR TABLE`, vous pouvez recevoir le message d'erreur FAILED : NullPointerException Name is null.

  Pour résoudre l'erreur, spécifiez une valeur pour l'TableInput`TableType`attribut dans le cadre de l'appel ou du AWS CloudFormation modèle d' AWS Glue `CreateTable`API. Parmi les valeurs possibles pour `TableType` figurent `EXTERNAL_TABLE` ou `VIRTUAL_VIEW`.

  Cette exigence s'applique uniquement lorsque vous créez une table à l'aide de l'opération AWS Glue `CreateTable` API ou du `AWS::Glue::Table` modèle. Si vous créez une table pour Athena en utilisant à l'aide d'une instruction DDL ou d'un crawler AWS Glue , la propriété `TableType` est définie pour vous automatiquement.

Fonctions prises en charge

Les fonctions prises en charge dans les requêtes Athena correspondent à celles de Trino et Presto. Pour plus d’informations sur les fonctions individuelles, veuillez consulter la section Fonctions et opérateurs de chaîne dans la documentation Trino ou Presto.

Les transformations de données transactionnelles ne sont pas prises en charge

Athena ne prend pas en charge les opérations basées sur les transactions (comme celles figurant dans Hive ou Presto) sur des données de table. Pour obtenir la liste complète des mots-clés non pris en charge, consultez DDL non pris en charge.

Les opérations qui modifient les états de table sont conformes à ACID

Lorsque vous créez, mettez à jour ou supprimez des tables, ces opérations sont garanties comme étant conformes à ACID. Par exemple, si plusieurs utilisateurs ou clients tentent de créer ou de modifier une table existante en même temps, un seule opération aboutira.

Les tables sont EXTERNAL

Sauf lors de la création de tables Iceberg, utilisez toujours le mot-clé `EXTERNAL`. Si vous utilisez `CREATE TABLE` sans le mot clé `EXTERNAL` pour des tables non Iceberg, Athena émet une erreur. Lorsque vous supprimez une table dans Athena, seules les métadonnées de la table sont supprimées ; les données sont conservées dans Simple Storage Service (Amazon S3).

Création de tables à l'aide AWS Glue de la console Athena

Vous pouvez créer des tables dans Athena en utilisant AWS Glue le formulaire d'ajout de table ou en exécutant une instruction DDL dans l'éditeur de requêtes Athena.

Pour créer une table à l'aide du AWS Glue robot

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans l'éditeur de requêtes, à côté de Tables and views (Tables et vues), choisissez Create (Créer) puis choisissez le Crawler AWS Glue .
3. Suivez les étapes sur la page Add crawler (Ajouter un Crawler) de la console AWS Glue pour ajouter un Crawler.

   Pour plus d'informations, consultez Utiliser des AWS Glue crawlers.

Création d'une table à l'aide du formulaire de création de table Athena

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans l'éditeur de requêtes, à côté de Tables and views (Tables et vues), choisissez Create (Créer) puis choisissez S3 bucket data (Données de compartiment S3).
3. Dans le formulaire Create Table From S3 bucket data (Créer une table à partir des données du compartiment S3), saisissez les informations pour créer votre table, puis choisissez Create table (Créer une table). Pour plus d'informations sur les champs du formulaire, consultez Ajout d'une table à l'aide d'un formulaire.

Pour créer une table à l'aide de Hive DDL

1. Dans le menu Database (Base de données), choisissez la base de données pour laquelle vous souhaitez créer une table. Si vous ne spécifiez pas de base de données dans votre instruction `CREATE TABLE`, la table est créée dans la base de données actuellement sélectionnée dans l'éditeur de requêtes.

2. Saisissez une instruction comme celle qui suit dans l'éditeur de requêtes, puis choisissez Run (Exécuter), ou appuyez sur **Ctrl+ENTER**.

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_logs (
    `Date` Date,
    Time STRING,
    Location STRING,
    Bytes INT,
    RequestIP STRING,
    Method STRING,
    Host STRING,
    Uri STRING,
    Status INT,
    Referrer STRING,
    OS String,
    Browser String,
    BrowserVersion String
) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
"input.regex" = "^(?!#)([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s
+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+[^\(]+[\(]([^\;]+).*\%20([^
\/]+)[\/](.*)$"
) LOCATION 's3://athena-examples-MyRegion/cloudfront/plaintext/';
```

Affichage des informations sur la table

Une fois que vous avez créé une table dans Athena, son nom s'affiche dans la liste Tables située sur la gauche. Choisissez les trois points verticaux à côté du nom de la table dans la console Athena pour afficher des informations sur la table et la gérer.

- Prévisualisation de la table – Affiche les 10 premières lignes de toutes les colonnes en exécutant l'instruction `SELECT * FROM "database_name"."table_name" LIMIT 10` dans l'éditeur de requêtes Athena.
- Génération du DDL de la table – Génère une instruction DDL que vous pouvez utiliser pour recréer la table en exécutant l'instruction `SHOW CREATE TABLE table_name` dans l'éditeur de requête Athena.
- Chargement de partitions – Exécute l'instruction `MSCK REPAIR TABLE table_name` dans l'éditeur de requêtes Athena. Cette option est disponible uniquement si la table comporte des partitions.

- Insert into editor (Insérer dans l'éditeur) – Insère le nom de la table dans l'éditeur de requêtes à l'emplacement de modification actuel.
- Suppression de table – Affiche une boîte de dialogue de confirmation vous demandant si vous souhaitez supprimer la table. Si vous êtes d'accord, cette option permet d'exécuter l'instruction `DROP TABLE table_name` dans l'éditeur de requêtes d'Athena.
- Propriétés de la table – Indique le nom de la table, le nom de la base de données, l'heure de création et si la table contient des données chiffrées.

Noms des tables, des bases de données et des colonnes

Suivez ces conseils pour nommer les objets de base de données dans Athena.

Exigences relatives aux noms de base de données, de tables et de colonnes

- Les caractères acceptables pour les noms de base de données, de tables et de colonnes AWS Glue doivent être une chaîne UTF-8. La longueur de la chaîne ne doit pas être inférieure à 1 ni supérieure à 255 octets. Le dépassement de cette limite génère une erreur telle que La valeur « name » ne remplit pas la condition : le membre doit avoir une longueur inférieure ou égale à 255. Les caractères pouvant être utilisés incluent des espaces et sont définis par le modèle de chaîne d'une seule ligne suivant :

```
[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\t]*
```

- Actuellement, le modèle AWS Glue regex permet d'ajouter des espaces de début au début des noms. Comme ces espaces de début peuvent être difficiles à détecter et peuvent entraîner des problèmes d'utilisabilité après leur création, évitez de créer des noms d'objets comportant des espaces de début.
- Si vous utilisez un AWS::Glue::Database AWS CloudFormation modèle pour créer une AWS Glue base de données et que vous ne spécifiez pas de nom de base de données, génère AWS Glue automatiquement un nom de base de données au format *resource_name–random_string* incompatible avec Athena.
- Vous pouvez utiliser le gestionnaire de AWS Glue catalogue pour renommer les colonnes, mais pas les noms de tables ou de bases de données. Pour contourner cette limitation, vous devez utiliser une définition de l'ancienne base de données pour créer une base de données portant le nouveau nom. Vous utilisez ensuite les définitions des tables de l'ancienne base de données pour recréer les tables de la nouvelle base de données. Pour ce faire, vous pouvez utiliser le AWS Glue

SDK AWS CLI ou. Pour les étapes, consultez Utilisation du AWS CLI pour recréer une AWS Glue base de données et ses tables.

Utilisation de minuscules pour les noms de tables et de colonnes de tables dans Athena

Athena accepte les majuscules et les minuscules dans les requêtes DDL et DML, mais les noms sont mis en minuscules au moment de l'exécution de la requête. Pour cette raison, évitez d'utiliser une casse mixte pour les noms de tables ou de colonnes et ne vous fiez pas à la casse seule dans Athena pour distinguer ces noms. Par exemple, si vous utilisez une instruction DDL pour créer une colonne nommée `Castle`, la colonne créée sera mise en minuscules à `castle`. Si vous spécifiez ensuite le nom de la colonne dans une requête DML comme `Castle` ou `CASTLE`, Athena mettra le nom en minuscules pour exécuter la requête, mais affichera l'en-tête de la colonne en utilisant la casse que vous avez choisie dans la requête.

Les noms de base de données, de tables et de colonnes doivent être inférieurs ou égaux à 255 caractères.

Noms commençant par un trait de soulignement

Lorsque vous créez des tables, utilisez des guillemets simples inversés pour entourer les noms de tables, de vues ou de colonnes qui commencent par un trait de soulignement. Par exemple :

```
CREATE EXTERNAL TABLE IF NOT EXISTS `_myunderscoretable`(
  `_id` string, `_index` string)
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
```

Noms de table, de vue ou de colonne commençant par des nombres

Lors de l'exécution de requêtes `SELECT`, `CTAS` ou `VIEW`, placez des guillemets autour d'identificateurs tels que les noms de table, de vue ou de colonne commençant par un chiffre. Par exemple :

```
CREATE OR REPLACE VIEW "123view" AS
SELECT "123columnone", "123columntwo"
FROM "234table"
```

Noms de colonnes et types complexes

Pour les types complexes, seuls les caractères alphanumériques, le trait de soulignement (_) et le point (.) sont autorisés dans les noms de colonnes. Pour créer une table et des mappages pour les clés qui ont des caractères restreints, vous pouvez utiliser une instruction DDL personnalisée. Pour plus d'informations, consultez l'article Création de tables dans Amazon Athena à partir de JSON imbriqué et de mappages à l'aide de JSON sur le AWS blog SerDe Big Data.

Mots réservés

Certains mots réservés dans Athena doivent être échappés. Pour spécifier des mots-clés réservés dans des instructions DDL, placez-les entre guillemets obliques (`). Pour spécifier des mots-clés réservés dans les instructions SQL SELECT et dans les requêtes sur les vues, placez-les entre des guillemets doubles (").

Pour plus d'informations, consultez Mots-clés réservés.

Ressources supplémentaires

Pour connaître la syntaxe complète de création de bases de données et de tables, consultez les pages suivantes.

- CREATE DATABASE
- CREATE TABLE

Pour plus d'informations sur les bases de données et les tables dans AWS Glue, consultez la section Bases de données et tables du manuel du AWS Glue développeur.

Mots-clés réservés

Lorsque vous exécutez des requêtes dans Athena qui incluent des mots-clés réservés, vous devez le spécifier en les plaçant entre caractères spéciaux. Utilisez les listes de cette rubrique pour vérifier quels mots-clés sont réservés dans Athena.

Pour spécifier des mots-clés réservés dans des instructions DDL, placez-les entre guillemets obliques (`). Pour spécifier des mots-clés réservés dans les instructions SQL SELECT et dans les requêtes sur les vues, placez-les entre des guillemets doubles (").

- Liste des mots-clés réservés dans des instructions DDL
- Liste des mots-clés réservés dans des instructions SQL SELECT

- Exemples de requêtes avec mots réservés

Liste des mots-clés réservés dans des instructions DDL

Athena utilise la liste suivante de mots-clés réservés dans ses instructions DDL. Si vous les utilisez sans les spécifier, Athena rencontre une erreur. Pour les spécifier, placez-les entre guillemets obliques (`).

Vous ne pouvez pas utiliser des mots-clés réservés DDL comme noms identificateurs dans des instructions DDL sans les placer entre guillemets obliques (`).

```
ALL, ALTER, AND, ARRAY, AS, AUTHORIZATION, BETWEEN, BIGINT,
BINARY, BOOLEAN, BOTH, BY, CASE, CASHE, CAST, CHAR, COLUMN,
CONF, CONSTRAINT, COMMIT, CREATE, CROSS, CUBE, CURRENT,
CURRENT_DATE, CURRENT_TIMESTAMP, CURSOR, DATABASE, DATE,
DAYOFWEEK, DECIMAL, DELETE, DESCRIBE, DISTINCT, DOUBLE, DROP,
ELSE, END, EXCHANGE, EXISTS, EXTENDED, EXTERNAL, EXTRACT,
FALSE, FETCH, FLOAT, FLOOR, FOLLOWING, FOR, FOREIGN, FROM,
FULL, FUNCTION, GRANT, GROUP, GROUPING, HAVING, IF, IMPORT,
IN, INNER, INSERT, INT, INTEGER, INTERSECT, INTERVAL, INTO,
IS, JOIN, LATERAL, LEFT, LESS, LIKE, LOCAL, MACRO, MAP, MORE,
NONE, NOT, NULL, NUMERIC, OF, ON, ONLY, OR, ORDER, OUT,
OUTER, OVER, PARTIALSCAN, PARTITION, PERCENT, PRECEDING,
PRECISION, PRESERVE, PRIMARY, PROCEDURE, RANGE, READS,
REDUCE, REGEXP, REFERENCES, REVOKE, RIGHT, RLIKE, ROLLBACK,
ROLLUP, ROW, ROWS, SELECT, SET, SMALLINT, START,TABLE,
TABLESAMPLE, THEN, TIME, TIMESTAMP, TO, TRANSFORM, TRIGGER,
TRUE, TRUNCATE, UNBOUNDED,UNION, UNIQUEJOIN, UPDATE, USER,
USING, UTC_TIMESTAMP, VALUES, VARCHAR, VIEWS, WHEN, WHERE,
WINDOW, WITH
```

Liste des mots-clés réservés dans des instructions SQL SELECT

Athena utilise la liste suivante de mots-clés réservés dans ses instructions SQL `SELECT` et les requêtes sur des vues.

Si vous utilisez ces mots-clés comme identificateurs, vous devez les placer entre guillemets doubles (") dans les instructions de votre requête.

```
ALTER, AND, AS, BETWEEN, BY, CASE, CAST, CONSTRAINT, CREATE,
```

```
CROSS, CUBE, CURRENT_CATALOG, CURRENT_DATE, CURRENT_PATH,
CURRENT_SCHEMA, CURRENT_TIME, CURRENT_TIMESTAMP, CURRENT_USER,
DEALLOCATE, DELETE, DESCRIBE, DISTINCT, DROP, ELSE, END, ESCAPE,
EXCEPT, EXECUTE, EXISTS, EXTRACT, FALSE, FIRST, FOR, FROM,
FULL, GROUP, GROUPING, HAVING, IN, INNER, INSERT, INTERSECT,
INTO, IS, JOIN, JSON_ARRAY, JSON_EXISTS, JSON_OBJECT,
JSON_QUERY, JSON_TABLE, JSON_VALUE, LAST, LEFT, LIKE,
LISTAGG, LOCALTIME, LOCALTIMESTAMP, NATURAL, NORMALIZE,
NOT, NULL, OF, ON, OR, ORDER, OUTER, PREPARE, RECURSIVE, RIGHT,
ROLLUP, SELECT, SKIP, TABLE, THEN, TRIM, TRUE, UESCAPE, UNION,
UNNEST, USING, VALUES, WHEN, WHERE, WITH
```

Exemples de requêtes avec mots réservés

La requête de l'exemple suivant utilise des accents graves (`) pour mettre en échappement les mots-clés réservés DDL partition et date qui sont utilisés pour un nom de table et l'un des noms de colonnes :

```
CREATE EXTERNAL TABLE `partition` (
`date` INT,
col2 STRING
)
PARTITIONED BY (year STRING)
STORED AS TEXTFILE
LOCATION 's3://DOC-EXAMPLE-BUCKET/test_examples/';
```

L'exemple de requête suivant inclut un nom de colonne contenant les mots-clés réservés DDL dans les instructions `ALTER TABLE ADD PARTITION` et les `ALTER TABLE DROP PARTITION`. Les mots-clés réservés DDL sont placés entre accents graves (`) :

```
ALTER TABLE test_table
ADD PARTITION (`date` = '2018-05-14')
```

```
ALTER TABLE test_table
DROP PARTITION (`partition` = 'test_partition_value')
```

Dans l'exemple suivant, la requête comprend un mot-clé réservé (fin) comme identificateur dans une instruction `SELECT`. Le mot-clé est placé entre guillemets droits :

```
SELECT *
```

```
FROM TestTable
WHERE "end" != nil;
```

Dans l'exemple suivant, la requête comprend un mot-clé réservé (en premier) comme identificateur dans une instruction `SELECT`. Le mot-clé est placé entre guillemets droits :

```
SELECT "itemId"."first"
FROM testTable
LIMIT 10;
```

Emplacement de table dans Simple Storage Service (Amazon S3)

Lorsque vous exécutez une `CREATE TABLE` requête dans Athéna, Athéna enregistre votre table dans le catalogue de AWS Glue données, dans lequel Athéna stocke vos métadonnées.

Pour spécifier le chemin d'accès à vos données dans Simple Storage Service (Amazon S3), utilisez la propriété `LOCATION`, comme illustré dans l'exemple suivant :

```
CREATE EXTERNAL TABLE `test_table`(
...
)
ROW FORMAT ...
STORED AS INPUTFORMAT ...
OUTPUTFORMAT ...
LOCATION s3://DOC-EXAMPLE-BUCKET/folder/
```

- Pour plus d'informations sur l'attribution de noms aux compartiments, consultez la section Restrictions et limites des compartiments du Guide de l'utilisateur Amazon Simple Storage Service.
- Pour plus d'informations sur l'utilisation des dossiers dans Simple Storage Service (Amazon S3), consultez la section Utilisation des dossiers du Guide de l'utilisateur de la console Amazon Simple Storage Service.

La propriété `LOCATION` dans Simple Storage Service (Amazon S3) spécifie tous les fichiers représentant votre table.

> **Important**
>
> Athena lit toutes les données stockées dans le dossier Simple Storage Service (Amazon S3) que vous spécifiez. Si vous avez des données que vous ne voulez pas qu'Athena lise,

ne les stockez pas dans le même dossier Simple Storage Service (Amazon S3) que les données que vous voulez qu'Athena lise. Si vous utilisez le partitionnement, pour qu'Athena analyse les données d'une partition, votre filtre `WHERE` doit inclure la partition. Pour plus d'informations, consultez Emplacement de table et partitions.

Lorsque vous spécifiez la propriété `LOCATION` dans l'instruction `CREATE TABLE`, suivez les instructions suivantes :

- Utilisez une barre oblique de fin.
- Vous pouvez utiliser un chemin d'accès à un dossier Simple Storage Service (Amazon S3) ou un alias de point d'accès Simple Storage Service (Amazon S3). Pour plus d'informations sur les alias de point d'accès Simple Storage Service (Amazon S3), consultez la rubrique Utilisation d'un alias de type compartiment pour votre point d'accès du Guide de l'utilisateur Simple Storage Service (Amazon S3).

Utilisez :

```
s3://DOC-EXAMPLE-BUCKET/folder/
```

```
s3://DOC-EXAMPLE-BUCKET-metadata-s3alias/folder/
```

N'utilisez aucun des éléments suivants pour spécifier `LOCATION` pour vos données.

- N'utilisez pas les noms de fichiers, les traits de soulignement, les caractères génériques ni les modèles glob pour spécifier les emplacements des fichiers.
- N'ajoutez pas la notation HTTP complète, telle que `s3.amazon.com`, au chemin d'accès au compartiment Simple Storage Service (Amazon S3).
- N'utilisez pas de dossiers vides comme `//` dans le chemin d'accès, comme ci-après : `S3://DOC-EXAMPLE-BUCKET/folder//folder/`. Bien qu'il s'agisse d'un chemin d'accès Simple Storage Service (Amazon S3), Athena ne l'autorise pas et le remplace par `s3://DOC-EXAMPLE-BUCKET/folder/folder/`, en supprimant la `/` supplémentaire.

  N'utilisez pas :

  ```
  s3://DOC-EXAMPLE-BUCKET
  s3://DOC-EXAMPLE-BUCKET/*
  ```

```
s3://DOC-EXAMPLE-BUCKET/mySpecialFile.dat
s3://DOC-EXAMPLE-BUCKET/prefix/filename.csv
s3://DOC-EXAMPLE-BUCKET.s3.amazon.com
S3://DOC-EXAMPLE-BUCKET/prefix//prefix/
arn:aws:s3:::DOC-EXAMPLE-BUCKET/prefix
s3://arn:aws:s3:<region>:<account_id>:accesspoint/<accesspointname>
https://<accesspointname>-<number>.s3-accesspoint.<region>.amazonaws.com
```

Emplacement de table et partitions

Vos données source peuvent être regroupées dans des dossiers Simple Storage Service (Amazon S3) appelés partitions en fonction d'un jeu de colonnes. Par exemple, ces colonnes peuvent représenter l'année, le mois et le jour de création de l'enregistrement spécifique.

Lorsque vous créez une table, vous pouvez choisir de la partitionner. Lorsque le service Athena exécute une requête SQL contre une table non partitionnée, il utilise la propriété `LOCATION` de la définition de la table comme chemin de base pour répertorier puis analyser tous les fichiers disponibles. Toutefois, avant de pouvoir interroger une table partitionnée, vous devez mettre à jour le catalogue de AWS Glue données avec les informations de partition. Cette information représente le schéma des fichiers dans la partition particulière et l'`LOCATION` des fichiers dans Simple Storage Service (Amazon S3) pour la partition.

- Pour savoir comment le AWS Glue robot ajoute des partitions, voir Comment un robot détermine-t-il à quel moment créer des partitions ? dans le Guide AWS Glue du développeur.
- Pour savoir comment configurer le crawler afin qu'il crée des tables pour les données dans les partitions existantes, consultez Utilisation de plusieurs sources de données avec les Crawlers.
- Vous pouvez également créer des partitions dans une table directement dans Athena. Pour plus d'informations, consultez Partitionnement de données dans Athena.

Lorsque le service Athena exécute une requête sur une table partitionnée, il vérifie si toutes les colonnes partitionnées sont utilisées dans la clause `WHERE` de la requête. Si des colonnes partitionnées sont utilisées, Athena demande AWS Glue au catalogue de données de renvoyer la spécification de partition correspondant aux colonnes de partition spécifiées. La spécification de la partition comprend la propriété `LOCATION` qui indique à Athena le préfixe Simple Storage Service (Amazon S3) à utiliser lors de la lecture des données. Dans ce cas, seules les données stockées dans ce préfixe sont analysées. Si vous n'utilisez pas de colonnes partitionnées dans la clause `WHERE`, Athena analyse tous les fichiers appartenant aux partitions de la table.

Pour des exemples d'utilisation du partitionnement avec Athena pour améliorer les performances des requêtes et réduire les coûts des requêtes, consultez les 10 meilleurs conseils d'optimisation des performances pour Amazon Athena.

Formats de stockage en colonnes

Apache Parquet et ORC sont des formats de stockage en colonnes optimisés pour une récupération rapide des données et utilisés dans AWS les applications analytiques.

Les formats de stockage en colonnes ont les caractéristiques suivantes qui les rendent adaptés à l'utilisation avec Athena :

- La compression par colonne, avec un algorithme de compression sélectionnée pour le type de données de la colonne pour économiser de l'espace de stockage dans Amazon S3 et réduire l'espace disque et d'I/O lors du traitement de la requête.
- Le prédicat pushdown dans Parquet et ORC permet aux requêtes Athena d'extraire uniquement les blocs dont elles ont besoin, améliorant ainsi les performances de requête. Lorsqu'une requête Athena obtient des valeurs de colonne spécifiques à partir de vos données, elle utilise les statistiques de prédicats de bloc de données, tels que les valeurs max/min, afin de déterminer s'il convient de lire ou d'ignorer le bloc.
- Le fractionnement des données dans Parquet et ORC permet à Athena de fractionner la lecture des données en plusieurs lecteurs et d'augmenter le parallélisme lors du traitement des requêtes.

Pour convertir vos données brutes existantes d'autres formats de stockage vers Parquet ou ORC, vous pouvez exécuter des requêtes CREATE TABLE AS SELECT (CTAS) dans Athena et spécifier un format de stockage de données tel que Parquet ou ORC, ou utiliser le Crawler. AWS Glue

Choix entre Parquet et ORC

Le choix entre ORC (Optimized Row Columnar) et Parquet dépend de vos besoins d'utilisation spécifiques.

Apache Parquet fournit des schémas de compression et de codage de données efficaces et convient parfaitement à l'exécution de requêtes complexes et au traitement de grandes quantités de données. Parquet est optimisé pour une utilisation avec Apache Arrow, ce qui peut être avantageux si vous utilisez des outils liés à Arrow.

ORC fournit un moyen efficace de stocker les données Hive. Les fichiers ORC sont souvent plus petits que les fichiers Parquet, et les index ORC peuvent accélérer les requêtes. De plus, ORC prend en charge les types complexes tels que les structures, les cartes et les listes.

Lorsque vous choisissez entre Parquet et ORC, prenez en considération les facteurs suivants :

Performances des requêtes : étant donné que Parquet prend en charge un plus large éventail de types de requêtes, Parquet peut s'avérer un meilleur choix si vous prévoyez d'effectuer des requêtes complexes.

Types de données complexes : si vous utilisez des types de données complexes, ORC peut se révéler un meilleur choix, car il prend en charge un plus large éventail de types de données complexes.

Taille des fichiers : si l'espace disque est un problème, ORC génère généralement des fichiers plus petits, ce qui peut réduire les coûts de stockage.

Compression : Parquet et ORC offrent tous deux une bonne compression, mais le format qui vous convient le mieux peut dépendre de votre cas d'utilisation spécifique.

Évolution : Parquet et ORC prennent tous deux en charge l'évolution du schéma, ce qui signifie que vous pouvez ajouter, supprimer ou modifier des colonnes au fil du temps.

Parquet et ORC sont tous deux de bons choix pour les applications de big data, mais tenez compte des exigences de votre scénario avant de choisir. Vous souhaiterez peut-être effectuer des points de référence sur vos données et vos requêtes afin de déterminer quel format convient le mieux à votre cas d'utilisation.

Conversion en formats de colonne

Les performances de vos requêtes Amazon Athena sont améliorées si vous convertissez les données dans des formats en colonne open source comme Apache parquet ou ORC.

Les options permettant de convertir facilement des données sources telles que JSON ou CSV en un format en colonnes incluent l'utilisation de requêtes CREATE TABLE AS ou l'exécution de tâches dans AWS Glue.

- Vous pouvez utiliser les requêtes `CREATE TABLE AS` (CTAS) pour convertir les données en Parquet ou ORC en une seule étape. Pour un exemple, consultez la section Exemple : écriture des résultats d'une requête dans un format différent sur la page Exemples de requêtes CTAS.

- Pour plus d'informations sur l'exécution AWS Glue d'une tâche visant à transformer des données CSV en Parquet, consultez la section « Transformer les données du format CSV au format Parquet » dans le billet de blog AWS Big Data Build a data lake AWS Glue foundation with Amazon S3. AWS Glue prend en charge l'utilisation de la même technique pour convertir les données CSV en ORC, ou les données JSON en Parquet ou ORC.

Partitionnement de données dans Athena

En partitionnant vos données, vous pouvez limiter la quantité de données analysées par chaque requête, ce qui améliore les performances et réduit les coûts. de n'importe quelle clé de partition. Il est d'usage de partitionner les données en fonction de l'heure, ce qui conduit souvent à un schéma de partitionnement à plusieurs niveaux. Par exemple, un client qui reçoit des données toutes les heures peut décider de les partitionner par année, par mois, par date et par heure. Un autre client, qui reçoit des données de nombreuses sources différentes mais qui sont chargées une seule fois par jour, peut les partitionner par date et par identifiant de source de données.

Athena peut utiliser des partitions de style Apache Hive dont les chemins de données contiennent des paires de valeur clé connectées par des signes égal (par exemple `country=us/...` ou `year=2021/month=01/day=26/...`). Ainsi, les chemins incluent à la fois les noms des clés de partition et les valeurs que chaque chemin représente. Pour charger de nouvelles partitions Hive dans une table partitionnée, vous pouvez utiliser la commande MSCK REPAIR TABLE qui fonctionne uniquement avec des partitions de style Hive.

Athena peut également utiliser des schémas de partitionnement non compatibles avec Hive. Par exemple, les CloudTrail journaux et les flux de diffusion Firehose utilisent des composants de chemin distincts pour les parties de date, telles que. `data/2021/01/26/us/6fc7845e.json` Pour les partitions de style non compatibles avec Hive, utilisez ALTER TABLE ADD PARTITION pour ajouter les partitions manuellement.

Considérations et restrictions

Lorsque vous utilisez le partitionnement, gardez à l'esprit les points suivants :

- Si vous interrogez une table partitionnée et spécifiez la partition dans la clause `WHERE`, Athena analyse les données uniquement à partir de cette partition. Pour plus d'informations, consultez Emplacement de table et partitions.
- Si vous exécutez des requêtes sur les compartiments Simple Storage Service (Amazon S3) avec un grand nombre d'objets et que les données ne sont pas partitionnées, de telles requêtes

peuvent affecter les limites de débit de demande GET dans Simple Storage Service (Amazon S3) et conduire à des exceptions Simple Storage Service (Amazon S3). Pour éviter des erreurs, partitionnez vos données. En outre, envisagez de régler vos taux de demande Simple Storage Service (Amazon S3). Pour plus d'informations, consultez la rubrique Bonnes pratiques de conception : optimisation des performances Simple Storage Service (Amazon S3).

- Les emplacements de partition à utiliser avec Athena doivent utiliser le protocole `s3` (par exemple, `s3://DOC-EXAMPLE-BUCKET/folder/`). Dans Athena, les emplacements qui utilisent d'autres protocoles (par exemple, `s3a://DOC-EXAMPLE-BUCKET/folder/`) provoquent des échecs de requête lorsque les requêtes `MSCK REPAIR TABLE` sont exécutées sur les tables contenant les données.
- Assurez-vous que le chemin Simple Storage Service (Amazon S3) est en minuscules et non en casse mixte (par exemple, `userid` au lieu de `userId`). Si le chemin S3 est en casse mixte, `MSCK REPAIR TABLE` n'ajoute pas les partitions au AWS Glue Data Catalog. Pour plus d'informations, consultez MSCK REPAIR TABLE.
- Étant donné que `MSCK REPAIR TABLE` analyse à la fois un dossier et ses sous-dossiers pour trouver un schéma de partition correspondant, veillez à conserver les données des différentes tables dans des hiérarchies de dossiers distinctes. Supposons, par exemple, que vous ayez des données pour le tableau 1 dans `s3://DOC-EXAMPLE-BUCKET1` et des données pour le tableau 2 dans`s3://DOC-EXAMPLE-BUCKET1/table-2-data`. Si les deux tables sont partitionnées par chaîne, `MSCK REPAIR TABLE` ajoutera les partitions de la table 2 à la table 1. Pour éviter cela, utilisez des structures de dossiers distinctes telles que `s3://DOC-EXAMPLE-BUCKET1` et à la `s3://DOC-EXAMPLE-BUCKET2` place. Notez que ce comportement est compatible avec Amazon EMR et Apache Hive.
- Si vous l'utilisez AWS Glue Data Catalog avec Athena, consultez la section AWS Glue Points de terminaison et quotas pour les quotas de service sur les partitions par compte et par table.
  - Bien qu'Athena prenne en charge l'interrogation de AWS Glue tables contenant 10 millions de partitions, Athena ne peut pas lire plus d'un million de partitions en un seul scan. Dans de tels scénarios, l'indexation des partitions peut s'avérer utile. Pour plus d'informations, consultez l'article du blog AWS Big Data Améliorez les performances des requêtes Amazon Athena à l'aide d'index de AWS Glue Data Catalog partition.
- Pour demander une augmentation du quota de partitions si vous utilisez le AWS Glue Data Catalog, visitez la console Service Quotas pour AWS Glue.

Création et chargement d'une table avec des données partitionnées

Pour créer une table qui utilise des partitions, utilisez la clause `PARTITIONED BY` dans votre instruction CREATE TABLE. La clause `PARTITIONED BY` définit les clés sur lesquelles les données doivent être partitionnées, comme dans l'exemple suivant. La clause `LOCATION` spécifie l'emplacement racine des données partitionnées.

```
CREATE EXTERNAL TABLE users (
first string,
last string,
username string
)
PARTITIONED BY (id string)
STORED AS parquet
LOCATION 's3://DOC-EXAMPLE-BUCKET'
```

Après avoir créé la table, vous chargez les données dans les partitions pour l'interrogation. Pour les partitions de style Hive, exécutez MSCK REPAIR TABLE. Pour les partitions de style non compatibles avec Hive, utilisez ALTER TABLE ADD PARTITION pour ajouter les partitions manuellement.

Préparation des données compatibles avec Hive et non compatibles avec Hive pour l'interrogation

Les sections suivantes expliquent comment préparer des données compatibles avec Hive et non compatibles avec Hive pour l'interrogation dans Athena.

Scénario 1 : données stockées sur Amazon S3 au format Hive

Les partitions sont stockées dans des dossiers distincts, dans Amazon S3. Par exemple, voici une liste partielle pour un exemple d'impressions publicitaires produites par la commande `aws s3 ls`, qui répertorie les objets S3 sous un préfixe spécifié :

```
aws s3 ls s3://elasticmapreduce/samples/hive-ads/tables/impressions/

    PRE dt=2009-04-12-13-00/
    PRE dt=2009-04-12-13-05/
    PRE dt=2009-04-12-13-10/
    PRE dt=2009-04-12-13-15/
    PRE dt=2009-04-12-13-20/
```

```
    PRE dt=2009-04-12-14-00/
    PRE dt=2009-04-12-14-05/
    PRE dt=2009-04-12-14-10/
    PRE dt=2009-04-12-14-15/
    PRE dt=2009-04-12-14-20/
    PRE dt=2009-04-12-15-00/
    PRE dt=2009-04-12-15-05/
```

Ici, les fichiers journaux sont stockés avec le nom de colonne (dt) défini sur les incréments de date, d'heures et de minutes. Lorsque vous fournissez une DDL avec l'emplacement du dossier parent, le schéma et le nom de la colonne partitionnée, Athena peut exécuter des requêtes sur les données figurant dans ces sous-dossiers.

Création de la table

Pour générer une table à partir de ces données, créez une partition avec 'dt' comme dans l'instruction DDL Athena suivante :

```
CREATE EXTERNAL TABLE impressions (
    requestBeginTime string,
    adId string,
    impressionId string,
    referrer string,
    userAgent string,
    userCookie string,
    ip string,
    number string,
    processId string,
    browserCookie string,
    requestEndTime string,
    timers struct<modelLookup:string, requestTime:string>,
    threadId string,
    hostname string,
    sessionId string)
PARTITIONED BY (dt string)
ROW FORMAT  serde 'org.apache.hive.hcatalog.data.JsonSerDe'
LOCATION 's3://elasticmapreduce/samples/hive-ads/tables/impressions/' ;
```

Cette table utilise le sérialiseur-désérialiseur JSON natif de Hive pour lire des données JSON stockées dans Simple Storage Service (Amazon S3). Pour de plus amples informations sur les formats pris en charge, veuillez consulter SerDe et formats de données pris en charge.

Exécuter MSCK REPAIR TABLE

Après avoir exécuté la requête `CREATE TABLE`, exécutez la commande `MSCK REPAIR TABLE` dans l'éditeur de requêtes Athena pour charger les partitions, comme dans l'exemple suivant.

```
MSCK REPAIR TABLE impressions
```

Une fois que vous avez exécuté cette commande, les données sont prêtes à être interrogées.

Exécution des requêtes de données

Exécutez des requêtes sur les données sur la table des impressions en utilisant la colonne de partition. Voici un exemple :

```
SELECT dt,impressionid FROM impressions WHERE dt<'2009-04-12-14-00' and
 dt>='2009-04-12-13-00' ORDER BY dt DESC LIMIT 100
```

Cette requête devrait montrer des résultats similaires aux suivants :

```
2009-04-12-13-20    ap3HcVKAWfXtgIPu6WpuUfAfL0DQEc
2009-04-12-13-20    17uchtodoS9kdeQP1x0XThKl5IuRsV
2009-04-12-13-20    JOUf1SCtRwviGw8sVcghqE5h0nkgtp
2009-04-12-13-20    NQ2XP0J0dvVbCXJ0pb4XvqJ5A4QxxH
2009-04-12-13-20    fFAItiBMsgqro9kRdIwbeX60SROaxr
2009-04-12-13-20    V4og4R9W6G3QjHHwF7gI1cSqig5D1G
2009-04-12-13-20    hPEPtBwk45msmwWTxPVVo1kVu4v11b
2009-04-12-13-20    v0SkfxegheD90gp31UCr6FplnKpx6i
2009-04-12-13-20    1iD9odVgOIi4QWkwHMcOhmwTkWDKfj
2009-04-12-13-20    b31tJiIA25CK8eDHQrHnbcknfSndUk
```

Scénario 2 : Les données ne sont pas partitionnées au format Hive

Dans l'exemple suivant, la commande `aws s3 ls` affiche les journaux ELB stockés dans Amazon S3. Notez que la mise en page des données n'utilise pas de paires `key=value` et n'est donc pas au format Hive. (L'option `--recursive` pour la commande `aws s3 ls` précise que tous les fichiers ou objets du répertoire ou du préfixe spécifié doivent être répertoriés.)

```
aws s3 ls s3://athena-examples-myregion/elb/plaintext/ --recursive

2016-11-23 17:54:46   11789573 elb/plaintext/2015/01/01/part-r-00000-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
```

```
2016-11-23 17:54:46    8776899 elb/plaintext/2015/01/01/part-r-00001-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:46    9309800 elb/plaintext/2015/01/01/part-r-00002-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:47    9412570 elb/plaintext/2015/01/01/part-r-00003-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:47   10725938 elb/plaintext/2015/01/01/part-r-00004-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:46    9439710 elb/plaintext/2015/01/01/part-r-00005-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:47          0 elb/plaintext/2015/01/01_$folder$
2016-11-23 17:54:47    9012723 elb/plaintext/2015/01/02/part-r-00006-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:47    7571816 elb/plaintext/2015/01/02/part-r-00007-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:47    9673393 elb/plaintext/2015/01/02/part-r-00008-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:48   11979218 elb/plaintext/2015/01/02/part-r-00009-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:48    9546833 elb/plaintext/2015/01/02/part-r-00010-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:48   10960865 elb/plaintext/2015/01/02/part-r-00011-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:48          0 elb/plaintext/2015/01/02_$folder$
2016-11-23 17:54:48   11360522 elb/plaintext/2015/01/03/part-r-00012-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:48   11211291 elb/plaintext/2015/01/03/part-r-00013-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:48    8633768 elb/plaintext/2015/01/03/part-r-00014-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:49   11891626 elb/plaintext/2015/01/03/part-r-00015-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:49    9173813 elb/plaintext/2015/01/03/part-r-00016-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:49   11899582 elb/plaintext/2015/01/03/part-r-00017-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:49          0 elb/plaintext/2015/01/03_$folder$
2016-11-23 17:54:50    8612843 elb/plaintext/2015/01/04/part-r-00018-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:50   10731284 elb/plaintext/2015/01/04/part-r-00019-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:50    9984735 elb/plaintext/2015/01/04/part-r-00020-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
```

```
2016-11-23 17:54:50    9290089 elb/plaintext/2015/01/04/part-r-00021-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:50    7896339 elb/plaintext/2015/01/04/part-r-00022-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:51    8321364 elb/plaintext/2015/01/04/part-r-00023-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:51          0 elb/plaintext/2015/01/04_$folder$
2016-11-23 17:54:51    7641062 elb/plaintext/2015/01/05/part-r-00024-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:51   10253377 elb/plaintext/2015/01/05/part-r-00025-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:51    8502765 elb/plaintext/2015/01/05/part-r-00026-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:51   11518464 elb/plaintext/2015/01/05/part-r-00027-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:51    7945189 elb/plaintext/2015/01/05/part-r-00028-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:51    7864475 elb/plaintext/2015/01/05/part-r-00029-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:51          0 elb/plaintext/2015/01/05_$folder$
2016-11-23 17:54:51   11342140 elb/plaintext/2015/01/06/part-r-00030-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:51    8063755 elb/plaintext/2015/01/06/part-r-00031-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:52    9387508 elb/plaintext/2015/01/06/part-r-00032-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:52    9732343 elb/plaintext/2015/01/06/part-r-00033-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:52   11510326 elb/plaintext/2015/01/06/part-r-00034-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:52    9148117 elb/plaintext/2015/01/06/part-r-00035-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:52          0 elb/plaintext/2015/01/06_$folder$
2016-11-23 17:54:52    8402024 elb/plaintext/2015/01/07/part-r-00036-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:52    8282860 elb/plaintext/2015/01/07/part-r-00037-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:52   11575283 elb/plaintext/2015/01/07/part-r-00038-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:53    8149059 elb/plaintext/2015/01/07/part-r-00039-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:53   10037269 elb/plaintext/2015/01/07/part-r-00040-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
```

```
2016-11-23 17:54:53   10019678 elb/plaintext/2015/01/07/part-r-00041-ce65fca5-
d6c6-40e6-b1f9-190cc4f93814.txt
2016-11-23 17:54:53          0 elb/plaintext/2015/01/07_$folder$
2016-11-23 17:54:53          0 elb/plaintext/2015/01_$folder$
2016-11-23 17:54:53          0 elb/plaintext/2015_$folder$
```

Exécuter ALTER TABLE ADD PARTITION

Étant donné que les données ne sont pas au format Hive, vous ne pouvez pas utiliser la commande `MSCK REPAIR TABLE` pour ajouter les partitions à la table une fois que vous l'avez créée. Au lieu de cela, vous pouvez utiliser la commande ALTER TABLE ADD PARTITION pour ajouter chaque partition manuellement. Par exemple, pour charger les données dans s3://athena-examples-*myregion*/elb/plaintext/2015/01/01/, vous pouvez exécuter la requête suivante : Notez qu'une colonne de partition distincte pour chaque dossier Simple Storage Service (Amazon S3) n'est pas requise et que la valeur clé de partition peut être différente de celle de la clé Simple Storage Service (Amazon S3).

```
ALTER TABLE elb_logs_raw_native_part ADD PARTITION (dt='2015-01-01') location 's3://
athena-examples-us-west-1/elb/plaintext/2015/01/01/'
```

Si une partition existe déjà, vous recevez l'erreur Partition already exists (Partition déjà existante). Pour éviter cette erreur, vous pouvez utiliser la clause `IF NOT EXISTS`. Pour plus d'informations, consultez ALTER TABLE ADD PARTITION. Pour supprimer une partition, vous pouvez utiliser ALTER TABLE DROP PARTITION.

Projection de partition

Pour éviter d'avoir à gérer les partitions, vous pouvez utiliser la projection de partition. La projection de partitions est une option pour les tables hautement partitionnées dont la structure est connue à l'avance. Dans une projection de partition, les valeurs et les emplacements de partition sont calculés à partir des propriétés de la table que vous configurez au lieu d'être lus à partir d'un référentiel de métadonnées. Étant donné que les calculs en mémoire sont plus rapides que la recherche à distance, l'utilisation de la projection de partition peut réduire considérablement les temps d'exécution des requêtes.

Pour plus d'informations, consultez Projection de partition avec Amazon Athena.

Ressources supplémentaires

- Pour plus d'informations sur les options de partitionnement des données Firehose, consultez. Exemple d'Amazon Data Firehose
- Vous pouvez automatiser l'ajout de partitions en utilisant le pilote JDBC.
- Vous pouvez utiliser CTAS et INSERT INTO pour partitionner un jeu de données. Pour plus d'informations, voir Utilisation de CTAS et INSERT INTO pour ETL et l'analyse des données.

Projection de partition avec Amazon Athena

Vous pouvez utiliser la projection de partition dans Athena pour accélérer le traitement des requêtes de tables hautement partitionnées et automatiser la gestion des partitions.

Dans une projection de partition, Athena calcule les valeurs et les emplacements de partition à l'aide des propriétés de table que vous configurez directement sur votre table dans AWS Glue. Les propriétés de table permettent à Athena de « projeter », ou de déterminer, les informations de partition nécessaires au lieu d'avoir à effectuer une recherche de métadonnées fastidieuse dans le AWS Glue Data Catalog. Étant donné que les opérations en mémoire sont souvent plus rapides que les opérations distantes, la projection de partition peut réduire le temps d'exécution des requêtes sur les tables hautement partitionnées. Selon les caractéristiques spécifiques de la requête et des données sous-jacentes, la projection de partition peut réduire considérablement le temps d'exécution des requêtes qui sont contraintes lors de la récupération des métadonnées de partition.

Élagage et projection pour les tables fortement partitionnées

L'élagage des partitions rassemble les métadonnées et les « élaguent » afin de ne conserver que les partitions qui s'appliquent à votre requête. Cette approche accélère souvent les requêtes. Athena utilise l'élagage de partition pour toutes les tables contenant des colonnes de partition, y compris les tables configurées pour la projection de partition.

Normalement, lors du traitement des requêtes, Athena fait un `GetPartitions` appel au AWS Glue Data Catalog avant de procéder à l'élagage de la partition. Si une table comporte un grand nombre de partitions, l'utilisation de `GetPartitions` peut nuire aux performances. Pour pallier ce problème, vous pouvez utiliser la projection de partition. La projection de partition permet à Athena d'éviter d'appeler `GetPartitions`, car la configuration de la projection de partition donne à Athena toutes les informations nécessaires pour créer les partitions.

Utilisation de la projection de partition

Pour utiliser la projection de partition, vous devez spécifier les plages de valeurs de partition et les types de projection pour chaque colonne de partition dans les propriétés de table du métastore Hive externe AWS Glue Data Catalog ou dans celui-ci. Ces propriétés personnalisées permettent à Athena d'identifier les modèles de partition prévus lorsqu'il exécute une requête au niveau de la table. Pendant l'exécution de la requête, Athena utilise ces informations pour projeter les valeurs de partition au lieu de les récupérer depuis le métastore Hive AWS Glue Data Catalog ou depuis un autre métastore Hive. Cette approche permet non seulement de réduire le temps d'exécution des requêtes, mais aussi d'automatiser la gestion des partitions, car cela élimine la nécessité de créer manuellement des partitions dans Athena, AWS Glue ou dans votre métastore Hive externe.

> **Important**
>
> L'activation de la projection de partition sur une table oblige Athena à ignorer toutes les métadonnées de partition enregistrées sur la table dans le métastore AWS Glue Data Catalog ou Hive.

Cas d’utilisation

Les scénarios dans lesquels la projection de partition est utile sont les suivants :

- Les requêtes relatives à une table hautement partitionnée ne se terminent pas aussi rapidement que vous le souhaitez.
- Vous ajoutez régulièrement des partitions aux tables au fur et à mesure que de nouvelles partitions de date ou d'heure sont créées dans vos données. Avec la projection de partition, vous configurez des plages de dates relatives qui peuvent être utilisées lors de l'arrivée de nouvelles données.
- Des données sont hautement partitionnées dans Simple Storage Service (Amazon S3). Les données ne sont pas pratiques à modéliser dans votre métastore AWS Glue Data Catalog ou dans Hive, et vos requêtes n'en lisent que de petites parties.

Structures de partition projetables

La projection de partition est plus facilement configurée lorsque vos partitions respectent un modèle prévisible comme suit (exemple non exhaustif) :

- Entiers : toute séquence continue d'entiers, telle que `[1, 2, 3, 4, ..., 1000]` ou `[0500, 0550, 0600, ..., 2500]`.
- Dates : toute séquence continue de dates ou de dates et heures, telle que `[20200101, 20200102, ..., 20201231]` ou `[1-1-2020 00:00:00, 1-1-2020 01:00:00, ..., 12-31-2020 23:00:00]`.
- Valeurs énumérées : ensemble fini de valeurs énumérées, telles que les codes d'aéroport ou. Régions AWS
- Service AWS journaux — Service AWS les journaux ont généralement une structure connue dont vous pouvez spécifier le schéma de partition AWS Glue et qu'Athena peut donc utiliser pour la projection de partitions.

Personnalisation du modèle de chemin de partition

Par défaut, Athena construit des emplacements de partition à l'aide du formulaire `s3://DOC-EXAMPLE-BUCKET/<table-root>/partition-col-1=<partition-col-1-val>/partition-col-2=<partition-col-2-val>/`, mais si vos données sont organisées différemment, Athena offre un mécanisme pour personnaliser ce modèle de chemin. Pour les étapes, consultez Spécification d'emplacements de stockage S3 personnalisés.

Considérations et restrictions

Les considérations suivantes s'appliquent :

- La projection de partition élimine le besoin de spécifier manuellement des partitions dans AWS Glue ou dans un métastore Hive externe.
- Lorsque vous activez la projection de partition sur une table, Athena ignore les métadonnées de partition présentes dans le métastore Hive AWS Glue Data Catalog ou dans un métastore Hive externe pour cette table.
- Si une partition projetée n'existe pas dans Simple Storage Service (Amazon S3), Athena projette quand même la partition. Athena ne renvoie pas d'erreur, mais aucune donnée n'est renvoyée. Toutefois, si un trop grand nombre de vos partitions sont vides, les performances peuvent être ralenties par rapport aux AWS Glue partitions traditionnelles. Si plus de la moitié de vos partitions projetées sont vides, il est recommandé d'utiliser des partitions traditionnelles.
- Les requêtes portant sur des valeurs qui dépassent les limites de l'intervalle défini pour la projection de partition ne renvoient pas d'erreur. Au lieu de cela, la requête s'exécute, mais retourne zéro ligne. Par exemple, si vous avez des données temporelles qui commencent en 2020

et qui sont définies comme `'projection.timestamp.range'='2020/01/01,NOW'`, une requête comme `SELECT * FROM table-name WHERE timestamp = '2019/02/02'` se terminera avec succès, mais renverra zéro ligne.

- La projection des partitions n'est utilisable que lorsque la table est interrogée par Athena. Si la même table est lue par un autre service tel que Amazon Redshift Spectrum, Athena pour Spark ou Amazon EMR, les métadonnées de partition standard sont utilisées.
- La projection de partition étant une fonctionnalité uniquement DML, `SHOW PARTITIONS` elle ne répertorie pas les partitions projetées par Athena mais non enregistrées dans le AWS Glue catalogue ou dans le métastore Hive externe.
- Athena n'utilise pas les propriétés des tables des vues comme configuration pour la projection des partitions. Pour contourner cette limitation, configurez et activez la projection de partition dans les propriétés des tables auxquelles les vues font référence.
- Les filtres de données Lake Formation ne peuvent pas être utilisés avec la projection de partitions dans la version 2 du moteur Athena.

Vidéo

La vidéo suivante montre comment utiliser la projection de partition pour augmenter les performances de vos requêtes dans Athena.

Projection de partition avec Amazon Athena

Rubriques

- Configuration de la projection de partition
- Types pris en charge pour la projection de partition
- Partitionnement d'ID dynamique
- Exemple d'Amazon Data Firehose

Configuration de la projection de partition

La configuration de la projection de partition dans les propriétés d'une table s'effectue en deux étapes :

1. Spécifiez les plages de données et les modèles pertinents pour chaque colonne de partition, ou utilisez un modèle personnalisé.

2. Activez la projection de partition pour la table.

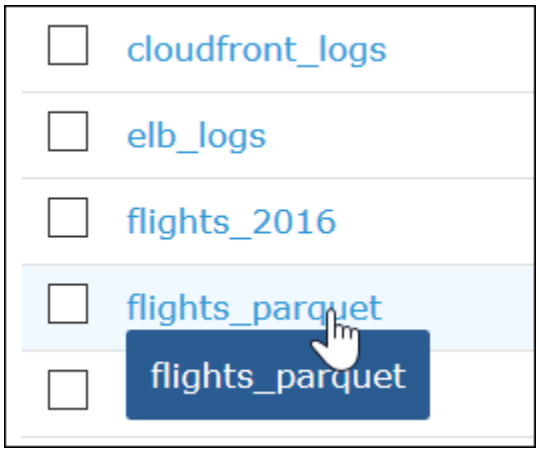
> **Note**
>
> Avant d'ajouter des propriétés de projection de partition à une table existante, la colonne de partition pour laquelle vous configurez les propriétés de projection de partition doit déjà exister dans le schéma de la table. Si la colonne de partition n'existe pas encore, vous devez ajouter manuellement une colonne de partition à la table existante. AWS Glue n'exécute pas cette étape automatiquement pour vous.

Cette section explique comment définir les propriétés de la table pour AWS Glue. Pour les définir, vous pouvez utiliser la AWS Glue console, les CREATE TABLE requêtes Athena ou AWS Glue APIles opérations. La procédure suivante indique comment définir les propriétés dans la AWS Glue console.

Pour configurer et activer la projection de partitions à l'aide de la AWS Glue console

1. Connectez-vous à la AWS Glue console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/glue/.
2. Choisissez l'onglet Tables.

   Sous l'onglet Tables, vous pouvez modifier des tables existantes ou choisir Add tables (Ajouter des tables) pour en créer de nouvelles. Pour plus d'informations sur l'ajout de tables manuellement ou à l'aide d'un Crawler, consultez la rubrique Travail avec des tables dans la console AWS Glue du Guide du développeur AWS Glue .
3. Dans la liste des tables, choisissez le lien correspondant à la table que vous souhaitez modifier.

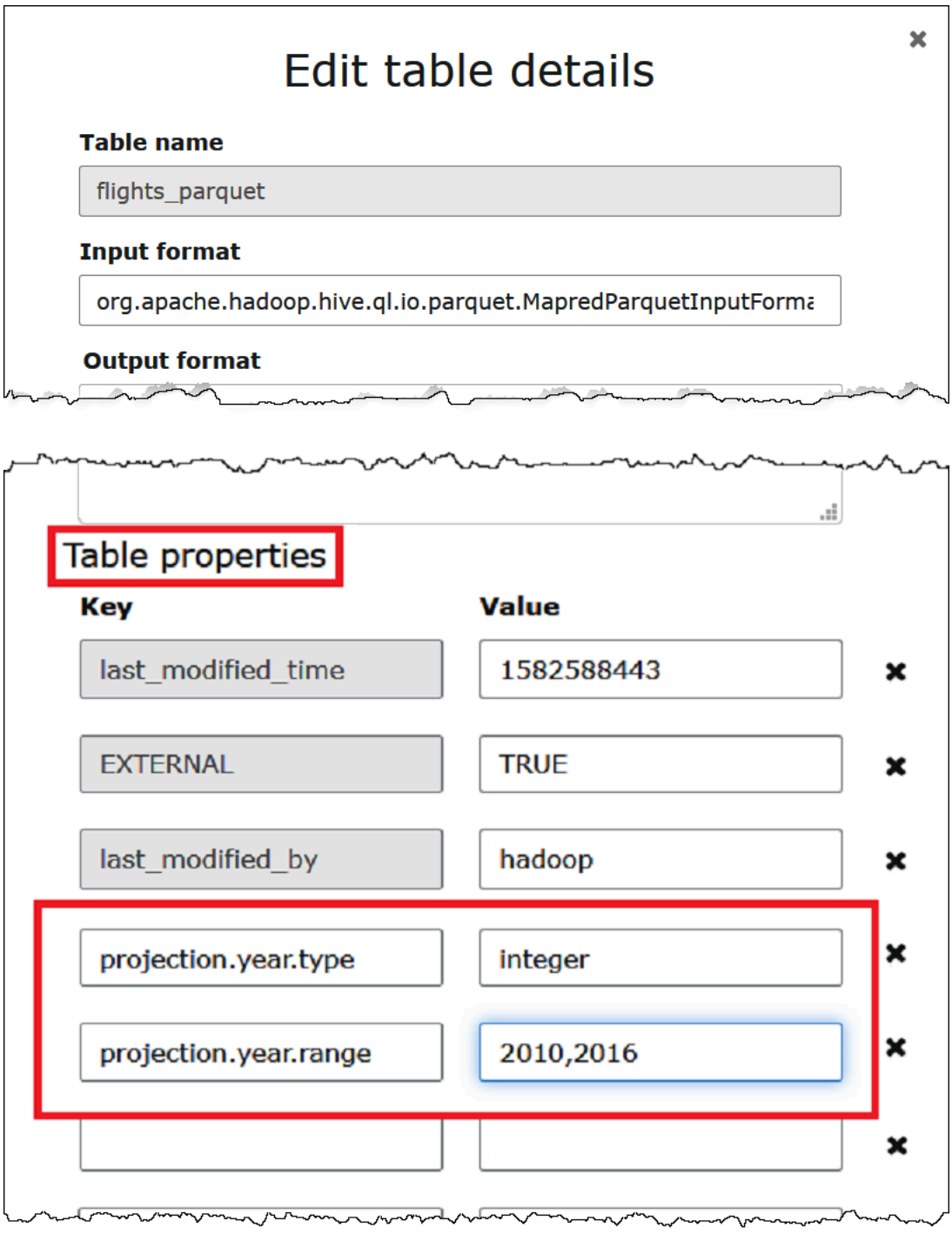


4. Sélectionnez Actions, puis Edit table (Modifier la table).
5. Sur la page Edit table (Modifier la table), dans la section Table properties (Propriétés de la table), ajoutez la paire clé-valeur suivante pour chaque colonne partitionnée :

   a. Pour Key (Clé), ajoutez `projection.`*`columnName`*`.type`.

   b. Pour Value (Valeur), ajoutez l'un des types pris en charge : `enum`, `integer`, `date`, ou `injected`. Pour plus d'informations, consultez Types pris en charge pour la projection de partition.
6. En suivant les instructions indiquées dans Types pris en charge pour la projection de partition, ajoutez des paires clé-valeur supplémentaires en fonction de vos exigences de configuration.

   L'exemple de configuration de table suivant configure la colonne `year` pour la projection de partition, limitant les valeurs susceptibles d'être renvoyées à une plage comprise entre 2010 et 2016.

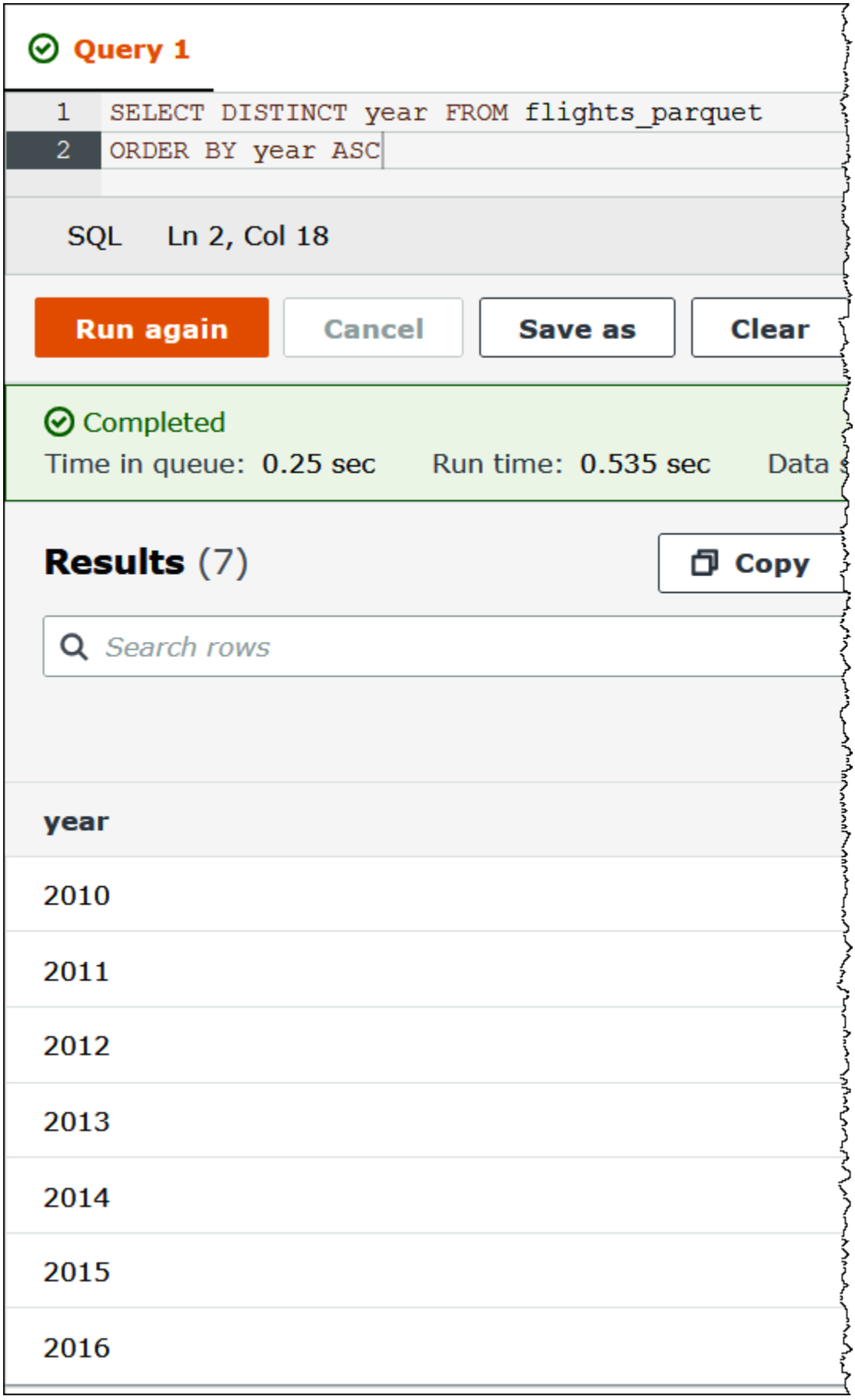


7. Ajoutez une paire clé-valeur pour activer la projection de partition. Pour Key (Clé), saisissez `projection.enabled` et pour Value (Valeur), saisissez `true`.

> **Note**
>
> Vous pouvez désactiver la projection de partition sur cette table à tout moment en définissant la valeur `projection.enabled` sur `false`.

8. Lorsque vous avez terminé, choisissez Save.
9. Dans l'éditeur de requête Athena, testez les colonnes que vous avez configurées pour la table.

   L'exemple de requête suivant utilise `SELECT DISTINCT` pour renvoyer les valeurs uniques de la colonne `year`. La base de données contient des données comprises entre la plage de dates 1987 et 2016, mais la propriété `projection.year.range` limite les valeurs renvoyées aux années 2010 à 2016.


Query 1
1 SELECT DISTINCT year FROM flights_parquet
2 ORDER BY year ASC
SQL Ln 2, Col 18
Run again
Cancel
Save as
Clear
Completed
Time in queue: 0.25 sec
Run time: 0.535 sec
Data
Results (7)
Copy
Search rows
year
2010
2011
2012
2013
2014
2015
2016

> **Note**
>
> Si vous définissez `projection.enabled` sur `true`, mais ne parvenez pas à configurer une ou plusieurs colonnes de partition, un message d'erreur du type suivant s'affiche : `HIVE_METASTORE_ERROR: Table database_name.table_name is configured for partition projection, but the following partition columns are missing projection configuration: [column_name] (table database_name.table_name).`

Spécification d'emplacements de stockage S3 personnalisés

Lorsque vous modifiez les propriétés d'une table dans AWS Glue, vous pouvez également spécifier un modèle de chemin Amazon S3 personnalisé pour les partitions projetées. Un modèle personnalisé permet à Athena de mapper correctement les valeurs de partition à des emplacements de fichiers Simple Storage Service (Amazon S3) personnalisés qui ne suivent pas un modèle `.../column=value/...` typique.

L'utilisation d'un modèle personnalisé est facultative. Toutefois, si vous utilisez un modèle personnalisé, celui-ci doit contenir un espace réservé pour chaque colonne de partition. Les emplacements modélisés doivent se terminer par une barre oblique afin que les fichiers de données partitionnés se trouvent dans un « dossier » par partition.

Pour spécifier un modèle d'emplacement de partition personnalisé

1. En suivant les étapes pour configurer et activer la projection de partition à l'aide de la AWS Glue console, ajoutez une paire clé-valeur supplémentaire qui spécifie un modèle personnalisé comme suit :

   a. Pour Key (Clé), saisissez `storage.location.template`.

   b. Pour Value (Valeur), spécifiez un emplacement qui inclut un espace réservé pour chaque colonne de partition. Assurez-vous que chaque espace réservé (et le chemin S3 lui-même) se termine par une barre oblique unique.

      Les exemples de valeurs de modèle suivants reposent sur une table avec les colonnes de partition `a`, `b` et `c`.

```
s3://DOC-EXAMPLE-BUCKET/table_root/a=${a}/${b}/some_static_subdirectory/${c}/
```

```
s3://DOC-EXAMPLE-BUCKET/table_root/c=${c}/${b}/some_static_subdirectory/${a}/
${b}/${c}/${c}/
```

Pour la même table, l'exemple de valeur de modèle suivant n'est pas valide, car il ne contient aucun espace réservé pour la colonne `c`.

```
s3://DOC-EXAMPLE-BUCKET/table_root/a=${a}/${b}/some_static_subdirectory/
```

2. Choisissez Appliquer.

Types pris en charge pour la projection de partition

Une table peut avoir n'importe quelle combinaison de types de colonnes de partition `enum`, `integer`, `date`, ou `injected`.

Type d'énumération

Utilisez le `enum` type pour les colonnes de partition dont les valeurs sont membres d'un ensemble énuméré (par exemple, des codes d'aéroport ou Régions AWS).

Définissez les propriétés de partition dans le tableau comme suit :

Nom de la propriété	Exemples de valeur	Description
`projection.` *`columnName`* `.type`	`enum`	Obligatoire. Type de projection à utiliser pour la colonne *`columnName`* . La valeur doit être `enum` (insensible à la casse) pour signaler l'utilisation du type énumération. L'espace de début et de fin est autorisé.

Nom de la propriété	Exemples de valeur	Description
`projection.` *`columnNam e`* `.values`	`A,B,C,D,E,F,G,Unkn own`	Obligatoire. Liste séparée par des virgules des valeurs de partition énumérées pour la colonne *`columnName`* . Tout espace est considéré comme faisant partie d'une valeur d'énumération.

> **Note**
>
> En tant que bonne pratique, nous recommandons de limiter l'utilisation des projections de partitions basées sur `enum` à quelques dizaines ou moins. Bien qu'il n'existe aucune limite spécifique pour les `enum` projections, la taille totale des métadonnées de votre table ne peut pas dépasser la AWS Glue limite d'environ 1 Mo lors de la compression gzip. Notez que cette limite est partagée entre les éléments clés de votre table, comme les noms de colonnes, l'emplacement, le format de stockage, etc. Si vous utilisez plus de quelques dizaines d'identifiants uniques dans votre projection `enum`, envisagez une autre approche, comme le compartimentage en un plus petit nombre de valeurs uniques dans un champ de substitution. En échangeant la cardinalité, vous pouvez contrôler le nombre de valeurs uniques dans votre champ `enum`.

Type d'entier

Utilisez le type entier pour les colonnes de partition dont les valeurs possibles peuvent être interprétées comme des entiers dans une plage définie. Les colonnes d'entier projetées sont actuellement limitées à la plage d'un entier Java long signé (-2^{63} à $2^{63}-1$ inclus).

Nom de la propriété	Exemples de valeur	Description
`projection.` *`columnNam e`* `.type`	`integer`	Obligatoire. Type de projectio n à utiliser pour la colonne *`columnName`* . La valeur doit être `integer` (insensible à la

Nom de la propriété	Exemples de valeur	Description
		casse) pour signaler l'utilisation du type entier. L'espace de début et de fin est autorisé.

`projection.` *`columnName`* `.range`	`0,10`
	
	`-1,8675309`
	
	`0001,9999`
		Obligatoire. Liste à deux éléments séparée par des virgules, qui fournit les valeurs de plage minimale et maximale à renvoyer par les requêtes dans la colonne *`columnName`* . Notez que les valeurs doivent être séparées par une virgule, et non par un tiret. Ces valeurs sont inclusives, peuvent être négatives et peuvent contenir des zéros de début. L'espace de début et de fin est autorisé.

`projection.` *`columnName`* `.interval`	`1`
	
	`5`
		Facultatif. Entier positif qui spécifie l'intervalle entre les valeurs de partition successives pour la colonne *`columnName`* . Par exemple, la valeur `range` « 1,3 » avec une valeur `interval` correspondant à « 1 » génère les valeurs 1, 2 et 3. La même valeur `range` avec une valeur `interval` correspondant à « 2 » génère les valeurs 1 et 3 et ignore 2. L'espace de début et de fin est autorisé. La valeur par défaut est 1.

Nom de la propriété	Exemples de valeur	Description

`projection.` *`columnNam e`* `.digits`	`1`
	
	`5`
		Facultatif. Entier positif spécifiant le nombre de chiffres à inclure dans la représentation finale de la valeur de partition pour la colonne *`columnName`* . Par exemple, la valeur `range` « 1,3 » avec une valeur `digits` correspondant à « 1 » génère les valeurs 1, 2 et 3. La même valeur `range` avec une valeur `digits` correspondant à « 2 » génère les valeurs 01, 02 et 03. L'espace de début et de fin est autorisé. La valeur par défaut est un nombre non statique de chiffres et ne contient aucun zéro de début.

Type de date

Utilisez le type de date pour les colonnes de partition dont les valeurs sont interprétables comme des dates (avec des heures facultatives) dans une plage définie.

> **Important**
>
> Les colonnes de date projetée sont générées en temps universel coordonné (UTC) au moment de l'exécution de la requête.

Nom de la propriété	Exemples de valeur	Description
`projection.` *`columnName`* `.type`	`date`	Obligatoire. Type de projection à utiliser pour la colonne *`columnName`* . La valeur doit être `date` (insensible à la casse) pour signaler l'utilisation du type date. L'espace de début et de fin est autorisé.

`projection.` *`columnName`* `.range`	`201701,201812`
	
	`01-01-2010,12-31-2018`
	
	`NOW-3YEARS,NOW`
	
	`201801,NOW+1MONTH`
		Obligatoire. Liste à deux éléments séparée par des virgules, qui fournit les valeurs `range` minimale et maximale à renvoyer par les requêtes dans la colonne *`columnName`* . Ces valeurs sont inclusives et peuvent utiliser n'importe quel format compatible avec les types de date Java `java.time.*` . Les valeurs minimales et maximales doivent utiliser le même format. Le format spécifié dans la propriété `.format` doit correspondre au format utilisé pour ces valeurs.
		
		Cette colonne peut également contenir des chaînes de date relatives, formatées dans ce modèle d'expression régulière :
		
		`\s*NOW\s*(([\+\-])\s*([0-9]+)\s*(YEARS?|MONTHS?|WEEKS?|DAYS?|HOURS?|MINUTES?|SECONDS?)\s*)?`
		
		Les espaces sont autorisés, mais les littéraux de type date sont considérés comme faisant partie des chaînes de date elles-mêmes.

`projection.` *`columnName`* `.format`	`yyyyMM`
	
	`dd-MM-yyyy`
		Obligatoire. Chaîne de format de date basée sur le format de date Java DateTimeFormatter. Il peut s'agir de n'importe quel type `Java.time.*` pris en charge.

Nom de la propriété	Exemples de valeur	Description
	`dd-MM-yyyy-HH-mm-ss`	

`projection.` *`columnName`* `.interval`	`1`
	
	`5`
		Entier positif qui spécifie l'intervalle entre les valeurs de partition successives pour la colonne *`columnName`* . Par exemple, la valeur `range 2017-01,2018-12` avec une valeur `interval` correspondant à 1 et une valeur `interval.unit` correspondant à `MONTHS` génère les valeurs 2017-01, 2017-02, 2017-03, etc. La même valeur `range` avec une valeur `interval` correspondant à 2 et une valeur `interval.unit` correspondant `MONTHS` génère les valeurs 2017-01, 2017-03, 2017-05, etc. L'espace de début et de fin est autorisé.
		
		Lorsque les dates fournies sont précises à un jour ou à un mois, la valeur `interval` est facultative, et la valeur par défaut est 1 jour ou 1 mois, respectivement. Dans le cas contraire, la valeur `interval` est obligatoire.

Nom de la propriété	Exemples de valeur	Description

`projection.` *`columnName`* `.interval.unit`	`YEARS`
	`MONTHS`
	`WEEKS`
	`DAYS`
	`HOURS`
	`MINUTES`
	`SECONDS`
	`MILLIS`
		Un mot d'unité de temps qui représente la forme sérialisée de a. ChronoUnit Les valeurs possibles sont `YEARS`, `MONTHS`, `WEEKS`, `DAYS`, `HOURS`, `MINUTES`, `SECONDS` ou `MILLIS`. Ces valeurs ne sont pas sensibles à la casse.
		
		Lorsque les dates fournies sont précises à un jour ou à un mois, la valeur `interval.unit` est facultative, et la valeur par défaut est 1 jour ou 1 mois, respectivement. Dans le cas contraire, la valeur `interval.unit` est obligatoire.

Type injecté

Utilisez le type injecté pour les colonnes de partition avec des valeurs possibles qui ne peuvent pas être générées de manière procédurale dans une plage logique, mais qui sont fournies dans la clause `WHERE` d'une requête en tant que valeur unique.

Il est important de garder à l'esprit les points suivants :

- Les requêtes au niveau des colonnes injectées échouent si aucune expression de filtre n'est fournie pour chaque colonne injectée.
- Les requêtes comportant plusieurs valeurs pour une expression de filtre sur une colonne injectée réussissent uniquement si les valeurs sont disjointes.
- Seules les colonnes de type `string` sont prises en charge.

Nom de la propriété	Valeu	Description
`projection.` *`columnName`* `.type`	`inje`	Obligatoire. Type de projection à utiliser pour la colonne *`columnName`* . Seul le type `string` est pris en charge.

Nom de la propriété	Valeu	Description
		La valeur spécifiée doit être `injected` (insensible à la casse). L'espace de début et de fin est autorisé.

Pour plus d’informations, consultez Utilisation du type de projection `injected`.

Partitionnement d'ID dynamique

Lorsque vos données sont partitionnées par une propriété avec une cardinalité élevée ou lorsque les valeurs ne peuvent pas être connues à l’avance, vous pouvez utiliser le type de projection `injected`. Les noms d’utilisateur et les ID d’appareils ou de produits sont des exemples de ce type de propriétés. Lorsque vous utilisez le type de projection `injected` pour configurer une clé de partition, Athena utilise les valeurs de la requête elle-même pour calculer l’ensemble des partitions qui seront lues.

Pour qu’Athena puisse exécuter une requête sur une table dont la clé de partition est configurée avec le type de projection `injected`, les critères suivants doivent être remplis :

- Votre requête doit inclure au moins une valeur pour la clé de partition.
- La ou les valeurs doivent être des littéraux ou des expressions qui peuvent être évalués sans lire aucune donnée.

Si l’un de ces critères n’est pas rempli, votre requête échoue avec l’erreur suivante :

CONSTRAINT_VIOLATION : Pour la colonne de partition projetée injectée *`column_name`*, la clause WHERE doit contenir uniquement des conditions d'égalité statiques, et au moins une de ces conditions doit être présente.

Utilisation du type de projection **`injected`**

Supposons que vous disposez d’un jeu de données composé d’événements provenant d’appareils IoT, partitionnés sur les ID des appareils. Ce jeu de données possède les caractéristiques suivantes :

- les ID d’appareil sont générés de manière aléatoire ;
- de nouveaux appareils sont fréquemment alloués ;
- il existe actuellement des centaines de milliers d’appareils, et dans le futur, il y en existera des millions.

Ce jeu de données est difficile à gérer en utilisant des métastores classiques. Il est difficile de synchroniser les partitions entre le stockage de données et le métastore, et le filtrage des partitions peut être lent lors de la planification des requêtes. Mais si vous configurez une table pour utiliser la projection de partition et que vous utilisez le type de projection `injected`, vous avez deux avantages : vous n'avez pas à gérer les partitions dans le métastore et vos requêtes n'ont pas à rechercher les métadonnées des partitions.

L'exemple `CREATE TABLE` suivant crée une table pour le jeu de données d'événements d'appareil que nous venons de décrire. La table utilise le type de projection injecté.

```
CREATE EXTERNAL TABLE device_events (
   event_time TIMESTAMP,
  data STRING,
  battery_level INT
)
PARTITIONED BY (
  device_id STRING
)
LOCATION "s3://DOC-EXAMPLE-BUCKET/prefix/"
TBLPROPERTIES (
  "projection.enabled" = "true",
  "projection.device_id.type" = "injected",
  "storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/prefix/${device_id}"
)
```

L'exemple de requête suivant recherche le nombre d'événements reçus par trois appareils spécifiques sur une période de 12 heures.

```
SELECT device_id, COUNT(*) AS events
FROM device_events
WHERE device_id IN (
  '4a770164-0392-4a41-8565-40ed8cec737e',
  'f71d12cf-f01f-4877-875d-128c23cbde17',
  '763421d8-b005-47c3-ba32-cc747ab32f9a'
)
AND event_time BETWEEN TIMESTAMP '2023-11-01 20:00' AND TIMESTAMP '2023-11-02 08:00'
GROUP BY device_id
```

Lorsque vous exécutez cette requête, Athena voit les trois valeurs de la clé de partition `device_id` et les utilise pour calculer les emplacements des partitions. Athena utilise la valeur de la propriété `storage.location.template` pour générer les emplacements suivants :

- s3://DOC-EXAMPLE-BUCKET/*prefix*/4a770164-0392-4a41-8565-40ed8cec737e
- s3://DOC-EXAMPLE-BUCKET/*prefix*/f71d12cf-f01f-4877-875d-128c23cbde17
- s3://DOC-EXAMPLE-BUCKET/*prefix*/763421d8-b005-47c3-ba32-cc747ab32f9a

Si vous omettez la propriété `storage.location.template` dans la configuration de projection de partition, Athena utilise le partitionnement de style Hive pour projeter les emplacements des partitions en fonction de la valeur contenue dans `LOCATION` (par exemple, `s3://DOC-EXAMPLE-BUCKET/prefix/device_id=4a770164-0392-4a41-8565-40ed8cec737e`).

Exemple d'Amazon Data Firehose

Lorsque vous utilisez Firehose pour transmettre des données à Amazon S3, la configuration par défaut écrit des objets avec des clés qui ressemblent à l'exemple suivant :

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/MM/dd/HH/file.extension
```

Pour créer une table Athena qui trouve les partitions automatiquement au moment de la requête, au lieu de devoir les ajouter à AWS Glue Data Catalog mesure que de nouvelles données arrivent, vous pouvez utiliser la projection de partitions.

L'`CREATE TABLE`exemple suivant utilise la configuration Firehose par défaut.

```
CREATE EXTERNAL TABLE my_ingested_data (
 ...
)
...
PARTITIONED BY (
 datehour STRING
)
LOCATION "s3://DOC-EXAMPLE-BUCKET/prefix/"
TBLPROPERTIES (
 "projection.enabled" = "true",
 "projection.datehour.type" = "date",
 "projection.datehour.format" = "yyyy/MM/dd/HH",
 "projection.datehour.range" = "2021/01/01/00,NOW",
 "projection.datehour.interval" = "1",
 "projection.datehour.interval.unit" = "HOURS",
 "storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/prefix/${datehour}/"
```

```
)
```

La clause `TBLPROPERTIES` dans l'instruction `CREATE TABLE` indique à Athena ce qui suit :

- Utiliser la projection de partition lors de l'interrogation de la table
- La clé de partition `datehour` est de type `date` (qui inclut une heure facultative)
- Comment les dates sont formatées
- La plage de dates et heures. Notez que les valeurs doivent être séparées par une virgule, et non par un tiret.
- Où trouver les données sur Amazon S3.

Lorsque vous interrogez la table, Athena calcule les valeurs pour `datehour` et utilise le modèle d'emplacement de stockage pour générer une liste d'emplacements de partition.

Utilisation du type **`date`**

Lorsque vous utilisez le type `date` pour une clé de partition projetée, vous devez spécifier une plage. Comme vous ne disposez d'aucune donnée concernant les dates antérieures à la création du flux de diffusion Firehose, vous pouvez utiliser la date de création comme point de départ. Et comme vous ne disposez pas de données pour les dates à l'avenir, vous pouvez utiliser le jeton spécial `NOW` comme fin.

Dans l'exemple `CREATE TABLE`, la date de début est spécifiée le 1er janvier 2021 à minuit UTC.

> **Note**
>
> Configurez une plage qui correspond aussi précisément que possible à vos données afin qu'Athena ne recherche que les partitions existantes.

Lorsqu'une requête est exécutée sur la table d'exemple, Athena utilise les conditions sur la clé de partition `datehour` en combinaison avec la plage pour générer des valeurs. Considérons la requête suivante :

```
SELECT *
FROM my_ingested_data
WHERE datehour >= '2020/12/15/00'
AND datehour < '2021/02/03/15'
```

La première condition dans la requête SELECT utilise une date antérieure au début de la plage de dates spécifiée par l'instruction CREATE TABLE. Étant donné que la configuration de projection de partitions ne spécifie aucune partition pour les dates antérieures au 1er janvier 2021, Athena recherche des données uniquement aux emplacements suivants et ignore les dates antérieures dans la requête.

```
s3://DOC-EXAMPLE-BUCKET/prefix/2021/01/01/00/
s3://DOC-EXAMPLE-BUCKET/prefix/2021/01/01/01/
s3://DOC-EXAMPLE-BUCKET/prefix/2021/01/01/02/
...
s3://DOC-EXAMPLE-BUCKET/prefix/2021/02/03/12/
s3://DOC-EXAMPLE-BUCKET/prefix/2021/02/03/13/
s3://DOC-EXAMPLE-BUCKET/prefix/2021/02/03/14/
```

De même, si la requête s'est exécutée à une date et une heure avant le 3 février 2021 à 15 h 00, la dernière partition refléterait la date et l'heure actuelles, et non la date et l'heure dans la condition de requête.

Si vous souhaitez rechercher les données les plus récentes, vous pouvez profiter du fait qu'Athena ne génère pas de dates futures et ne spécifier qu'une `datehour` de début, comme dans l'exemple suivant.

```
SELECT *
FROM my_ingested_data
WHERE datehour >= '2021/11/09/00'
```

Choix des clés de partition

Vous pouvez spécifier comment la projection de partition mappe les emplacements des partitions aux clés de partition. Dans l'exemple CREATE TABLE de la section précédente, la date et l'heure ont été combinées dans une clé de partition appelée datehour, mais d'autres schémas sont possibles. Par exemple, vous pouvez également configurer une table avec des clés de partition distinctes pour l'année, le mois, le jour et l'heure.

Cependant, si vous divisez les dates en année, mois et jour, le type de projection de partition de `date` ne peut pas être utilisé. Une alternative consiste à séparer la date de l'heure afin de continuer à exploiter le type de projection de partition de `date`, tout en facilitant la lecture des requêtes qui spécifient des plages d'heures.

Dans cette optique, l'exemple `CREATE TABLE` suivant sépare la date de l'heure. Étant donné que `date` est un mot réservé dans SQL, l'exemple utilise `day` comme nom de la clé de partition qui représente la date.

```
CREATE EXTERNAL TABLE my_ingested_data2 (
 ...
)
...
PARTITIONED BY (
 day STRING,
 hour INT
)
LOCATION "s3://DOC-EXAMPLE-BUCKET/prefix/"
TBLPROPERTIES (
 "projection.enabled" = "true",
 "projection.day.type" = "date",
 "projection.day.format" = "yyyy/MM/dd",
 "projection.day.range" = "2021/01/01,NOW",
 "projection.day.interval" = "1",
 "projection.day.interval.unit" = "DAYS",
 "projection.hour.type" = "integer",
 "projection.hour.range" = "0,23",
 "projection.hour.digits" = "2",
 "storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/prefix/${day}/${hour}/"
)
```

Dans l'exemple d'instruction `CREATE TABLE`, l'heure est une clé de partition distincte, configurée comme un nombre entier. La configuration de la clé de partition des heures spécifie la plage 0 à 23, et que l'heure doit être formatée à deux chiffres lorsque Athena génère les emplacements de partition.

Une requête pour la table `my_ingested_data2` pourrait ressembler à ceci :

```
SELECT *
FROM my_ingested_data2
WHERE day = '2021/11/09'
AND hour > 3
```

Types de clés de partition et types de projection de partitions

Notez que la clé `datehour` dans le premier exemple `CREATE TABLE` est configurée comme `date` dans la configuration de projection de partition, mais le type de clé de partition est `string`. Il en va de même pour `day` dans le second exemple. Les types de la configuration de projection de partition indiquent uniquement à Athena comment formater les valeurs lorsqu'elle génère les emplacements de partition. Les types que vous spécifiez ne modifient pas le type de clé de partition. Dans les requêtes, `datehour` et `day` sont de type `string`.

Lorsqu'une requête inclut une condition telle que `day = '2021/11/09'`, Athena analyse la chaîne située à droite de l'expression en utilisant le format de date spécifié dans la configuration de projection de partition. Après avoir vérifié que la date est comprise dans la plage configurée, Athena utilise à nouveau le format de date pour insérer la date sous forme de chaîne dans le modèle d'emplacement de stockage.

De même, pour une condition de requête telle que `day > '2021/11/09'`, Athena analyse le côté droit et génère une liste de toutes les dates correspondantes dans la plage configurée. Le format de date est ensuite utilisé pour insérer chaque date dans le modèle d'emplacement de stockage afin de créer la liste des emplacements de partition.

Écrire la même condition que `day > '2021-11-09'` ou `day > DATE '2021-11-09'` ne fonctionne pas. Dans le premier cas, le format de date ne correspond pas (notez les traits d'union plutôt que les barres obliques) et, dans le second cas, les types de données ne correspondent pas.

Utilisation de préfixes personnalisés et de partitionnement dynamique

Firehose peut être configuré avec des préfixes personnalisés et un partitionnement dynamique. À l'aide de ces fonctions, vous pouvez configurer les clés Amazon S3 et configurer des schémas de partitionnement qui prennent mieux en charge votre cas d'utilisation. Vous pouvez également utiliser la projection de partitions avec ces schémas de partitionnement et les configurer en conséquence.

Par exemple, vous pouvez utiliser la fonction de préfixe personnalisé pour obtenir des clés Amazon S3 dont les dates sont au format ISO au lieu du schéma par défaut `yyyy/MM/dd/HH`.

Vous pouvez également combiner des préfixes personnalisés avec un partitionnement dynamique pour extraire une propriété, comme dans les messages `customer_id` Firehose, comme dans l'exemple suivant.

```
prefix/!{timestamp:yyyy}-!{timestamp:MM}-!{timestamp:dd}/!
{partitionKeyFromQuery:customer_id}/
```

Avec ce préfixe Amazon S3, le flux de diffusion Firehose écrirait des objets sur des clés telles que. `s3://DOC-EXAMPLE-BUCKET/prefix/2021-11-01/customer-1234/file.extension` Pour une propriété comme `customer_id`, où les valeurs peuvent ne pas être connues à l'avance, vous pouvez utiliser le type de projection de partition `injected` et utilisez une instruction `CREATE TABLE` comme suit :

```
CREATE EXTERNAL TABLE my_ingested_data3 (
 ...
)
...
PARTITIONED BY (
 day STRING,
 customer_id STRING
)
LOCATION "s3://DOC-EXAMPLE-BUCKET/prefix/"
TBLPROPERTIES (
 "projection.enabled" = "true",
 "projection.day.type" = "date",
 "projection.day.format" = "yyyy-MM-dd",
 "projection.day.range" = "2021-01-01,NOW",
 "projection.day.interval" = "1",
 "projection.day.interval.unit" = "DAYS",
 "projection.customer_id.type" = "injected",
 "storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/prefix/${day}/${customer_id}/"
)
```

Lorsque vous interrogez une table comportant une clé de partition de type `injected`, votre requête doit inclure une valeur pour cette clé de partition. Une requête pour la table `my_ingested_data3` pourrait ressembler à ceci :

```
SELECT *
FROM my_ingested_data3
WHERE day BETWEEN '2021-11-01' AND '2021-11-30'
AND customer_id = 'customer-1234'
```

Dates au format ISO

Comme les valeurs de la clé de partition `day` sont au format ISO, vous pouvez également utiliser le type `DATE` pour la clé de partition day au lieu de `STRING`, comme dans l'exemple suivant :

```
PARTITIONED BY (day DATE, customer_id STRING)
```

Lorsque vous effectuez une requête, cette stratégie vous permet d'utiliser des fonctions de date sur la clé de partition sans analyse ni diffusion, comme dans l'exemple suivant :

```
SELECT *
FROM my_ingested_data3
WHERE day > CURRENT_DATE - INTERVAL '7' DAY
AND customer_id = 'customer-1234'
```

> **Note**
>
> La spécification d'une clé de partition du type `DATE` suppose que vous avez utilisé la fonctionnalité de préfixe personnalisé pour créer des clés Amazon S3 dont les dates sont au format ISO. Si vous utilisez le format Firehose par défaut de`yyyy/MM/dd/HH`, vous devez spécifier la clé de partition en tant que type, `string` même si la propriété de table correspondante est de type`date`, comme dans l'exemple suivant :
>
> ```
> PARTITIONED BY (
>   `mydate` string)
> TBLPROPERTIES (
>   'projection.enabled'='true',
>    ...
>   'projection.mydate.type'='date',
>   'storage.location.template'='s3://DOC-EXAMPLE-BUCKET/prefix/${mydate}')
> ```

Création d'une table à partir des résultats des requêtes (CTAS)

Une requête `CREATE TABLE AS SELECT` (CTAS) crée une nouvelle table dans Athena à partir des résultats d'une instruction `SELECT` d'une autre requête. Athena stocke les fichiers de données créés par l'instruction ETC dans un emplacement spécifié dans Simple Storage Service (Amazon S3). Pour la syntaxe, consultez CREATE TABLE AS.

`CREATE TABLE AS` combine une instruction DDL `CREATE TABLE` avec une instruction DML `SELECT` et contient donc techniquement tant des instructions DDL que DML. Notez toutefois qu'à des fins de Service Quotas, les requêtes CTAS dans Athena sont traitées comme des requêtes DML. Pour plus d'informations sur les Service Quotas Athena, consultez Service Quotas.

Utilisez les requêtes CTAS pour :

- Créer des tables à partir des résultats de requêtes dans une étape, sans interroger plusieurs fois des ensembles de données brutes. Il est ainsi plus facile de travailler avec des ensembles de données brutes.
- Transformer les résultats des requêtes et faire migrer les tables vers d'autres formats de table tels qu'Apache Iceberg. Cela améliore les performances de requête et réduit les coûts de requête dans Athena. Pour plus d'informations, veuillez consulter Création de tables Iceberg.
- Transformer les résultats des requêtes en formats de stockage tels que Parquet et ORC. Cela améliore les performances de requête et réduit les coûts de requête dans Athena. Pour plus d'informations, veuillez consulter Formats de stockage en colonnes.
- Créer des copies de tables existantes qui contiennent uniquement les données dont vous avez besoin.

Rubriques

Considérations et limitations relatives aux requêtes CTAS

Les sections suivantes décrivent les considérations et les limites à prendre en compte lorsque vous utilisez des requêtes (CTAS) `CREATE TABLE AS SELECT` dans Athena.

Syntaxe de requête CTAS

La syntaxe de requête CTAS diffère de la syntaxe de `CREATE [EXTERNAL] TABLE` utilisée pour créer des tables. veuillez consulter CREATE TABLE AS.

Comparaison entre requêtes CTAS et vues

Les requêtes CTAS écrivent de nouvelles données dans un emplacement spécifié dans Simple Storage Service (Amazon S3), tandis que les vues n'écrivent pas de données.

Emplacement des résultats de requête CTAS

Si votre groupe de travail remplace le paramètre côté client pour l'emplacement des résultats de requête, Athena crée votre table à cet emplacement `s3://DOC-EXAMPLE-BUCKET/tables/<query-id>/`. Pour afficher l'emplacement des résultats de la requête spécifié pour le groupe de travail, affichez les détails du groupe de travail.

Si votre groupe de travail ne remplace pas l'emplacement des résultats de la requête, vous pouvez utiliser la syntaxe `WITH (external_location ='s3://DOC-EXAMPLE-BUCKET/')` de votre requête CTAS pour spécifier l'emplacement où les résultats de vos requêtes CTAS sont stockés.

> **Note**
>
> La propriété `external_location` doit spécifier un emplacement vide. Une requête CTAS vérifie que l'emplacement de chemin (préfixe) dans le compartiment est vide et ne remplace jamais les données si l'emplacement en contient déjà. Pour utiliser à nouveau le même emplacement, supprimez les données dans l'emplacement du préfixe de clé dans le compartiment.

Si vous omettez la syntaxe `external_location` et que vous n'utilisez pas le paramètre de groupe de travail, Athena utilise votre paramètre côté client pour l'emplacement des résultats de la requête et crée votre table à cet emplacement `s3://DOC-EXAMPLE-BUCKET/<Unsaved-or-query-name>/<year>/<month/<date>/tables/<query-id>/`.

Localisation de fichiers orphelins

Si une instruction `CTAS` ou `INSERT INTO` échoue, il est possible que les données orphelines soient laissées dans l'emplacement des données. Étant donné qu'Athena ne supprime pas, dans certains cas, les données ou les données partielles de votre compartiment, vous pourrez être en mesure de lire ces données partielles dans les requêtes suivantes. Pour localiser les fichiers orphelins en vue d'une inspection ou d'une suppression, vous pouvez utiliser le fichier manifeste de données fourni par Athena pour suivre la liste des fichiers à écrire. Pour plus d'informations, consultez Identification des fichiers de sortie de requête et DataManifestLocation.

Clauses ORDER BY ignorées

Dans une requête CTAS, Athena ignore les clauses `ORDER BY` contenues dans la partie `SELECT` de la requête.

Selon la spécification SQL (ISO 9075, partie 2), l'ordre des lignes d'une table spécifiée par une expression de requête n'est garanti que pour l'expression de requête qui contient immédiatement la clause `ORDER BY`. Les tables en SQL sont dans tous les cas intrinsèquement désordonnées, et l'implémentation de`ORDER BY` dans les clauses de sous-requête entraînerait à la fois une baisse des performances de la requête et n'entraînerait pas une sortie ordonnée. Ainsi, dans les requêtes Athena CTAS, rien ne garantit que l'ordre spécifié par la clause `ORDER BY` sera préservé lors de l'écriture des données.

Formats de stockage des résultats de requête

Par défaut, les résultats de requête CTAS sont stockés dans Parquet, si vous ne spécifiez pas de format de stockage des données. Vous pouvez stocker les résultats CTAS dans `PARQUET`, `ORC`, `AVRO`, `JSON` et `TEXTFILE`. Les délimiteurs à plusieurs caractères ne sont pas pris en charge pour le format `TEXTFILE` CTAS. Les requêtes CTAS ne nécessitent pas de spécifier a SerDe pour interpréter les transformations de format. veuillez consulter Example: Writing query results to a different format.

Formats de compression

La compression `GZIP` est utilisée pour les résultats des requêtes CTAS aux formats JSON et TEXTFILE. Pour Parquet, vous pouvez utiliser `GZIP` ou `SNAPPY`, la valeur par défaut étant `GZIP`. Pour Parquet, vous pouvez utiliser `LZ4`, `SNAPPY`, `ZLIB` ou `ZSTD`, la valeur par défaut étant `ZLIB`. Pour obtenir des exemples CTAS qui spécifient une compression, consultez Example: Specifying data storage and compression formats. Pour plus d'informations sur la compression dans Athena, consultez Prise en charge de la compression Athena.

Limites de partition et de compartiment

Vous pouvez partitionner et compartimenter les données de résultats d'une requête CTAS. Pour plus d'informations, consultez Partitionnement et compartimentation dans Athena. Lors de la création d'une table partitionnée à l'aide du CTAS, Athena a une limite d'écriture de 100 partitions.

Incluez les prédicats de partitionnement et de mise en compartiments à la fin de la clause `WITH` qui spécifie les propriétés de la table de destination. Pour plus d'informations, consultez Example: Creating bucketed and partitioned tables et Partitionnement et compartimentation dans Athena.

Pour plus d'informations sur le contournement de la limitation de 100 partitions, consultez Utilisation de CTAS et de INSERT INTO pour contourner la limite de 100 partitions.

Chiffrement

Vous pouvez chiffrer les résultats de requête CTAS dans Simple Storage Service (Amazon S3), comme vous le faites pour chiffrer les autres résultats de requête dans Athena. Pour plus d'informations, consultez Chiffrement des résultats des requêtes Athena stockées dans Simple Storage Service (Amazon S3).

Propriétaire du compartiment attendu

Pour les instructions CTAS, le paramètre de propriétaire du compartiment attendu ne s'applique pas à l'emplacement de la table de destination dans Simple Storage Service (Amazon S3). Le paramètre de propriétaire du compartiment attendu s'applique uniquement à l'emplacement de sortie Simple Storage Service (Amazon S3) que vous spécifiez pour les résultats de la requête Athena. Pour plus d'informations, consultez Spécification d'un emplacement de résultats de requête à l'aide de la console Athena.

Types de données

Les types de données de colonne pour une requête CTAS sont les mêmes que celles spécifiées pour la requête d'origine.

Exécution de requêtes CTAS dans la console

Dans la console Athena, vous pouvez créer une requête CTAS à partir d'une autre requête.

Pour créer une requête CTAS à partir d'une autre requête

1. Exécutez la requête dans l'éditeur de requêtes de la console Athena.
2. Au bas de l'éditeur de requêtes, choisissez l'option Create (Créer), puis choisissez Table from query (Table à partir d'une requête).
3. Dans le formulaire Create table as select (Créer une table en tant que sélection), remplissez les champs comme suit :

    a. Pour Table name (Nom de la table), saisissez le nom de votre nouvelle table. Utilisez uniquement des minuscules et des traits de soulignement, par exemple `my_select_query_parquet`.

    b. Pour Database configuration (Configuration de base de données), utilisez les options pour choisir une base de données existante ou en créer une.

c. (Facultatif) Dans Result configuration (Configuration des résultats), pour Location of CTAS query results (Emplacement des résultats de requête CTAS), si le paramètre d'emplacement des résultats de votre requête de groupe de travail ne remplace pas cette option, effectuez l'une des opérations suivantes :

   - Saisissez le chemin d'un emplacement S3 existant dans la zone de recherche, ou choisissez Browse S3 (Parcourir S3) pour choisir un emplacement dans une liste.
   - Choisissez View (Afficher) pour ouvrir la page Buckets (Compartiments) de la console Amazon S3 où vous pouvez consulter plus d'informations sur vos compartiments existants et choisir ou créer un compartiment avec vos propres paramètres.

   Vous devez spécifier un emplacement vide dans Amazon S3 où les données seront produites. Si des données existent déjà dans l'emplacement que vous spécifiez, la requête échoue avec une erreur.

   Si le paramètre d'emplacement de résultats de votre requête de groupe de travail remplace ce paramètre d'emplacement, Athena crée votre table dans l'emplacement `s3://DOC-EXAMPLE-BUCKET/tables/query_id/`

d. Pour Data format (Format de données), spécifiez le format dans lequel se trouvent vos données.

   - Table type (Type de table) : le type de table par défaut dans Athena est Apache Hive.
   - File format (Format de fichier) : choisissez parmi des options telles que CSV, TSV, JSON, Parquet ou ORC. Pour plus d'informations sur les formats Parquet et ORC, consultez Formats de stockage en colonnes.
   - Write compression (Compression d'écriture) : (facultatif) choisissez un format de compression. Athena prend en charge divers formats de compression pour la lecture et l'écriture de données, y compris la lecture d'une table qui utilise plusieurs formats de compression. Par exemple, Athena peut lire avec succès les données d'une table qui utilise le format de fichier Parquet lorsque certains fichiers Parquet sont compressés avec Snappy et d'autres fichiers Parquet sont compressés avec GZIP. Le même principe s'applique aux formats de stockage ORC, fichier texte et JSON. Pour plus d'informations, consultez Prise en charge de la compression Athena.
   - Partitions : (facultatif) sélectionnez les colonnes que vous souhaitez partitionner. Le partitionnement de vos données limite la quantité de données analysées par chaque

requête, ce qui améliore les performances et réduit les coûts. de n'importe quelle clé de partition. Pour plus d'informations, consultez Partitionnement de données dans Athena.

- Buckets (Compartiments) : (facultatif) sélectionnez les colonnes que vous souhaitez mettre en compartiments. La mise en compartiments est une technique qui regroupe les données en fonction de colonnes spécifiques au sein d'une même partition. Ces colonnes sont appelées clés de compartiment. En regroupant les données connexes dans un seul compartiment (un fichier au sein d'une partition), vous réduisez considérablement la quantité de données scannées par Athena, améliorant ainsi les performances des requêtes et réduisant les coûts. Pour plus d'informations, consultez Partitionnement et compartimentation dans Athena.

e. Pour Preview table query (Requête de prévisualisation de table), vérifiez votre requête. Pour la syntaxe de requête, consultez CREATE TABLE AS.

f. Choisissez Créer un tableau.

Pour créer une requête CTAS à l'aide d'un modèle SQL

Utilisez le modèle `CREATE TABLE AS SELECT` pour créer une requête CTAS dans l'éditeur de requêtes.

1. Dans la console Athena, près de Tables and views (Tables et vues), choisissez Create table (Créer une table) puis CREATE TABLE AS SELECT. Cela remplit l'éditeur de requêtes avec une requête CTAS avec des valeurs d'espace réservé.
2. Dans l'éditeur de requêtes, modifiez la requête si nécessaire. Pour la syntaxe de requête, consultez CREATE TABLE AS.
3. Cliquez sur Run (Exécuter).

Pour obtenir des exemples, consultez Exemples de requêtes CTAS.

Partitionnement et compartimentation dans Athena

Le partitionnement et la compartimentation sont deux moyens de réduire la quantité de données qu'Athena doit analyser lorsque vous exécutez une requête. Le partitionnement et la compartimentation sont complémentaires et peuvent être utilisés ensemble. La réduction de la quantité de données analysées permet d'améliorer les performances et de réduire les coûts. Pour des instructions générales sur les performances des requêtes Athena, consultez 10 meilleurs conseils de réglage des performances pour Amazon Athena.

Qu'est-ce que le partitionnement ?

Le partitionnement consiste à organiser les données dans des répertoires (ou « préfixes ») sur Amazon S3 en fonction d'une propriété particulière des données. Ces propriétés sont appelées clés de partition. Une clé de partition courante est la date ou une autre unité de temps telle que l'année ou le mois. Cependant, un jeu de données peut être partitionné par plusieurs clés. Par exemple, les données relatives aux ventes de produits peuvent être partitionnées par date, catégorie de produit et marché.

Choix du mode de partitionnement

Les propriétés qui sont toujours ou fréquemment utilisées dans les requêtes et qui ont une faible cardinalité peuvent être utilisées comme clés de partition. Il y a un compromis entre le fait d'avoir trop de partitions et d'en avoir trop peu. Lorsque le nombre de partitions est trop élevé, l'augmentation du nombre de fichiers entraîne une surcharge. Le filtrage des partitions elles-mêmes entraîne également une certaine surcharge. Lorsque le nombre de partitions est trop faible, les requêtes doivent souvent analyser davantage de données.

Création d'une table partitionnée

Lorsqu'un jeu de données est partitionné, vous pouvez créer une table partitionnée dans Athena. Une table partitionnée est une table qui possède des clés de partition. Lorsque vous utilisez `CREATE TABLE`, vous ajoutez des partitions à la table. Lorsque vous utilisez `CREATE TABLE AS`, les partitions créées sur Amazon S3 sont automatiquement ajoutées à la table.

Dans une instruction `CREATE TABLE`, vous spécifiez les clés de partition dans la clause `PARTITIONED BY (`*`column_name data_type`*`)`. Dans une instruction `CREATE TABLE AS`, vous spécifiez les clés de partition dans une clause `WITH (partitioned_by = ARRAY['`*`partition_key`*`'])` ou `WITH (partitioning = ARRAY['`*`partition_key`*`'])` pour les tables Iceberg. Pour des raisons de performances, les clés de partition doivent toujours être de type `STRING`. Pour plus d'informations, consultez Utiliser STRING comme type de données pour les clés de partition.

Pour des informations de syntaxe `CREATE TABLE` et `CREATE TABLE AS` supplémentaires, consultez CREATE TABLE et Propriétés de la table CTAS.

Interrogation de tables partitionnées

Lorsque vous interrogez une table partitionnée, Athena utilise les prédicats de la requête pour filtrer la liste des partitions. Il utilise ensuite les emplacements des partitions correspondantes pour traiter

les fichiers trouvés. Athena peut réduire efficacement la quantité de données analysées en ne lisant simplement pas les données contenues dans les partitions qui ne correspondent pas aux prédicats de la requête.

Exemples

Supposons que vous ayez une table partitionnée par `sales_date` et `product_category` et que vous souhaitiez connaître le chiffre d'affaires total sur une semaine dans une catégorie spécifique. Vous incluez des prédicats dans les colonnes `sales_date` et `product_category` pour vous assurer qu'Athena analyse uniquement la quantité minimale de données, comme dans l'exemple suivant.

```
SELECT SUM(amount) AS total_revenue
FROM sales
WHERE sales_date BETWEEN '2023-02-27' AND '2023-03-05'
AND product_category = 'Toys'
```

Supposons que vous disposiez d'un jeu de données partitionné par date mais également doté d'un horodatage précis.

Avec les tables Iceberg, vous pouvez déclarer qu'une clé de partition a une relation avec une colonne, mais avec les tables Hive, le moteur de requête n'a aucune connaissance des relations entre les colonnes et les clés de partition. Pour cette raison, vous devez inclure un prédicat à la fois sur la colonne et sur la clé de partition de votre requête afin de vous assurer que celle-ci n'analyse pas plus de données que nécessaire.

Supposons, par exemple, que la table `sales` de l'exemple précédent comporte également une colonne `sold_at` du type de données `TIMESTAMP`. Si vous souhaitez obtenir le chiffre d'affaires uniquement pour une période spécifique, vous devez écrire la requête comme suit :

```
SELECT SUM(amount) AS total_revenue
FROM sales
WHERE sales_date = '2023-02-28'
AND sold_at BETWEEN TIMESTAMP '2023-02-28 10:00:00' AND TIMESTAMP '2023-02-28
 12:00:00'
AND product_category = 'Toys'
```

Pour plus d'informations sur cette différence entre l'interrogation des tables Hive et Iceberg, consultez Comment écrire des requêtes pour des champs d'horodatage qui sont également partitionnés dans le temps.

Qu'est-ce que la compartimentation ?

La compartimentation est une méthode pour organiser les enregistrements d'un jeu de données en catégories appelées compartiments.

La présente signification des termes compartiment et compartimentation diffère de celle des compartiments Amazon S3 et ne doit pas être confondue avec celle-ci. Lors de la compartimentation des données, les enregistrements ayant la même valeur pour une propriété sont placés dans le même compartiment. Les enregistrements sont répartis aussi uniformément que possible entre les compartiments afin que chaque compartiment contienne à peu près la même quantité de données.

Dans la pratique, les compartiments sont des fichiers, et une fonction de hachage détermine le compartiment dans lequel un enregistrement est placé. Un jeu de données compartimenté comportera un ou plusieurs fichiers par compartiment et par partition. Le compartiment auquel appartient un fichier est codé dans le nom du fichier.

Avantages de la compartimentation

La compartimentation est utile lorsqu'un jeu de données est compartimenté par une certaine propriété et que vous souhaitez récupérer des enregistrements dans lesquels cette propriété possède une certaine valeur. Comme les données sont compartimentées, Athena peut utiliser la valeur pour déterminer les fichiers à consulter. Supposons, par exemple, qu'un jeu de données soit compartimenté par `customer_id` et que vous souhaitiez rechercher tous les enregistrements d'un client spécifique. Athena détermine le compartiment qui contient ces enregistrements et ne lit que les fichiers qu'il contient.

Les colonnes présentant une cardinalité élevée (c'est-à-dire comportant de nombreuses valeurs distinctes), qui sont distribuées de manière uniforme et que vous interrogez fréquemment concernant des valeurs spécifiques se prêtent bien à la compartimentation.

> **Note**
>
> Athena ne prend pas en charge l'utilisation `INSERT INTO` pour ajouter de nouveaux enregistrements à des tables compartimentées.

Types de données pris en charge pour le filtrage sur les colonnes compartimentées

Vous pouvez ajouter des filtres sur des colonnes compartimentées contenant certains types de données. Athena prend en charge le filtrage uniquement sur les colonnes compartimentées avec les types de données suivants :

- BOOLEAN
- BYTE
- DATE
- DOUBLE
- FLOAT
- INT
- LONG
- SHORT
- CHAÎNE
- VARCHAR

Prise en charge de Hive et Spark

La version 2 du moteur Athena prend en charge les jeux de données compartimentés à l'aide de l'algorithme de compartiment Hive, et la version 3 du moteur Athena prend également en charge l'algorithme de compartimentation Apache Spark. La compartimentation Hive est le méthode par défaut. Si votre jeu de données est compartimenté à l'aide de l'algorithme Spark, utilisez la clause `TBLPROPERTIES` pour définir la valeur de la propriété `bucketing_format` sur `spark`.

> **Note**
>
> Athena a une limite de 100 partitions dans une requête `CREATE TABLE AS SELECT` (CTAS). De même, vous pouvez ajouter un maximum de 100 partitions à une table de destination avec une instruction INSERT INTO. Cette limite de 100 ne s'applique que lorsque la table est compartimentée et partitionnée.
> Si vous dépassez cette limite, le message d'erreur HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets (Dépassement de la limite de 100 rédacteurs ouverts pour les partitions/compartiments) peut s'afficher. Pour contourner ces limitations, vous pouvez utiliser une instruction CTAS et une série d'instructions `INSERT`

INTO qui créent ou insèrent jusqu'à 100 partitions chacune. Pour plus d'informations, consultez Utilisation de CTAS et de INSERT INTO pour contourner la limite de 100 partitions.

Exemple de compartimentation CREATE TABLE

Pour créer une table pour un jeu de données compartimenté existant, utilisez la clause CLUSTERED BY (*column*) suivie de la clause INTO *N* BUCKETS. La clause INTO *N* BUCKETS spécifie le nombre de compartiments dans lesquels les données sont compartimentées.

Dans l'exemple CREATE TABLE suivant, le jeu de données sales est compartimenté par customer_id en 8 compartiments à l'aide de l'algorithme Spark. L'instruction CREATE TABLE utilise les clauses CLUSTERED BY et TBLPROPERTIES pour définir les propriétés en conséquence.

```
CREATE EXTERNAL TABLE sales (...)
...
CLUSTERED BY (`customer_id`) INTO 8 BUCKETS
...
TBLPROPERTIES (
  'bucketing_format' = 'spark'
)
```

Exemple de compartimentation CREATE TABLE AS (CTAS)

Pour spécifier la compartimentation avec CREATE TABLE AS, utilisez les paramètres bucketed_by et bucket_count, comme dans l'exemple suivant.

```
CREATE TABLE sales
WITH (
  ...
  bucketed_by = ARRAY['customer_id'],
  bucket_count = 8
)
AS SELECT ...
```

Exemple de requête de compartimentation

L'exemple de requête suivant recherche les noms de produits qu'un client spécifique a achetés au cours d'une semaine.

```
SELECT DISTINCT product_name
```

```
FROM sales
WHERE sales_date BETWEEN '2023-02-27' AND '2023-03-05'
AND customer_id = 'c123'
```

Si cette table est partitionnée par `sales_date` et compartimentée par `customer_id`, Athena peut calculer le compartiment dans lequel se trouvent les enregistrements du client. Athena lit tout au plus un fichier par partition.

Ressources supplémentaires

- Pour un exemple `CREATE TABLE AS` qui crée des tables tant partitionnées que compartimentées, consultez Exemple : création de tables partitionnées et compartimentées.
- Pour plus d'informations sur la mise en œuvre du AWS cloisonnement sur les lacs de données, notamment à l'aide d'une instruction Athena CTAS AWS Glue , pour Apache Spark, et du partitionnement pour les tables Apache Iceberg, consultez AWS le billet de blog consacré au Big Data Optimize data layout by buketing with Amazon Athena and to accelerating queries in aval. AWS Glue

Exemples de requêtes CTAS

Utilisez les exemples suivants pour créer des requêtes CTAS. Pour obtenir des informations sur la syntaxe CTAS, consultez CREATE TABLE AS.

Dans cette section :

- Example: Duplicating a table by selecting all columns
- Example: Selecting specific columns from one or more tables
- Example: Creating an empty copy of an existing table
- Example: Specifying data storage and compression formats
- Example: Writing query results to a different format
- Example: Creating unpartitioned tables
- Example: Creating partitioned tables
- Example: Creating bucketed and partitioned tables
- Example: Creating an Iceberg table with Parquet data
- Example: Creating an Iceberg table with Avro data

Example Exemple : duplication d'une table en sélectionnant toutes les colonnes

L'exemple suivant crée une table en copiant toutes les colonnes d'une table :

```
CREATE TABLE new_table AS
SELECT *
FROM old_table;
```

Dans la variation suivante du même exemple, votre instruction `SELECT` inclut également une clause `WHERE`. Dans ce cas, la requête sélectionne uniquement les lignes du tableau qui respectent la clause `WHERE` :

```
CREATE TABLE new_table AS
SELECT *
FROM old_table
WHERE condition;
```

Example Exemple : sélection de colonnes spécifiques à partir d'une ou plusieurs tables

L'exemple suivant crée une nouvelle requête qui s'exécute sur un ensemble de colonnes à partir d'une autre table :

```
CREATE TABLE new_table AS
SELECT column_1, column_2, ... column_n
FROM old_table;
```

Cette variante du même exemple crée une nouvelle table à partir de colonnes spécifiques provenant de plusieurs tables :

```
CREATE TABLE new_table AS
SELECT column_1, column_2, ... column_n
FROM old_table_1, old_table_2, ... old_table_n;
```

Example Exemple : création d'une copie vide d'une table existante

L'exemple suivant utilise `WITH NO DATA` pour créer une nouvelle table qui est vide et a le même schéma que la table d'origine :

```
CREATE TABLE new_table
AS SELECT *
```

```
FROM old_table
WITH NO DATA;
```

Example Exemple : spécification de formats de stockage et de compression des données

Grâce à CTAS, vous pouvez utiliser une table source dans un format de stockage afin de créer une autre table dans un format de stockage différent.

Utilisation de la propriété du `format` pour spécifier `ORC`, `PARQUET`, `AVRO`, `JSON` ou `TEXTFILE` comme format de stockage pour la nouvelle table.

Pour les formats de stockage `PARQUET`, `ORC`, `TEXTFILE` et `JSON`, utilisez la propriété `write_compression` pour spécifier le format de compression des données de la nouvelle table. Pour plus d'informations sur les formats de compression pris en charge par chaque format de fichier, consultez Prise en charge de la compression Athena.

L'exemple suivant indique que les données de la table `new_table` sont stockées au format Parquet et utilisent la compression Snappy. La compression par défaut pour Parquet est `GZIP`.

```
CREATE TABLE new_table
WITH (
      format = 'Parquet',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table;
```

L'exemple suivant indique que les données de la table `new_table` sont stockées au format ORC et utilisent la compression Snappy. La compression par défaut pour ORC est ZLIB.

```
CREATE TABLE new_table
WITH (format = 'ORC',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table ;
```

L'exemple suivant indique que les données de la table `new_table` sont stockées au format de fichier texte et utilisent la compression Snappy. La compression par défaut pour les formats de fichier texte et JSON est GZIP.

```
CREATE TABLE new_table
WITH (format = 'TEXTFILE',
```

```
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table ;
```

Example Exemple : écriture des résultats d'une requête dans un format différent

La requête CTAS suivante sélectionne tous les enregistrements depuis `old_table`, qui peuvent être stockés au format CSV ou dans un autre format, et crée une nouvelle table avec les données sous-jacentes enregistrées dans Simple Storage Service (Amazon S3) au format ORC :

```
CREATE TABLE my_orc_ctas_table
WITH (
      external_location = 's3://DOC-EXAMPLE-BUCKET/my_orc_stas_table/',
      format = 'ORC')
AS SELECT *
FROM old_table;
```

Example Exemple : création de tables non partitionnées

Les exemples suivants créent des tables qui ne sont pas partitionnées. Les données de table sont stockées dans des formats différents. Certains de ces exemples spécifient l'emplacement externe.

L'exemple suivant crée une requête CTAS qui stocke les résultats sous la forme d'un fichier texte :

```
CREATE TABLE ctas_csv_unpartitioned
WITH (
     format = 'TEXTFILE',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_csv_unpartitioned/')
AS SELECT key1, name1, address1, comment1
FROM table1;
```

Dans l'exemple suivant, les résultats sont stockés dans Parquet et l'emplacement par défaut des résultats est utilisé :

```
CREATE TABLE ctas_parquet_unpartitioned
WITH (format = 'PARQUET')
AS SELECT key1, name1, comment1
FROM table1;
```

Dans la requête suivante, la table est stockée au format JSON, et des colonnes spécifiques sont sélectionnées à partir des résultats de la table d'origine :

```
CREATE TABLE ctas_json_unpartitioned
WITH (
     format = 'JSON',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_json_unpartitioned/')
AS SELECT key1, name1, address1, comment1
FROM table1;
```

Dans l'exemple suivant, le format est ORC :

```
CREATE TABLE ctas_orc_unpartitioned
WITH (
     format = 'ORC')
AS SELECT key1, name1, comment1
FROM table1;
```

Dans l'exemple suivant, le format est Avro :

```
CREATE TABLE ctas_avro_unpartitioned
WITH (
     format = 'AVRO',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_avro_unpartitioned/')
AS SELECT key1, name1, comment1
FROM table1;
```

Example Exemple : création de tables partitionnées

Les exemples suivants illustrent des requêtes `CREATE TABLE AS SELECT` pour les tables partitionnées dans différents formats de stockage, en utilisant `partitioned_by` et d'autres propriétés dans la clause `WITH`. Pour la syntaxe, consultez Propriétés de la table CTAS. Pour plus d'informations sur le choix de colonnes pour le partitionnement, consultez Partitionnement et compartimentation dans Athena.

> **Note**
>
> Répertoriez les colonnes de partition à la fin de la liste de colonnes dans l'instruction `SELECT`. Vous pouvez partitionner par plusieurs colonnes et avoir jusqu'à 100 combinaisons partition-compartiment uniques. Par exemple, vous pouvez avoir 100 partitions si aucun compartiment n'est spécifié.

```
CREATE TABLE ctas_csv_partitioned
WITH (
     format = 'TEXTFILE',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_csv_partitioned/',
     partitioned_by = ARRAY['key1'])
AS SELECT name1, address1, comment1, key1
FROM tables1;
```

```
CREATE TABLE ctas_json_partitioned
WITH (
     format = 'JSON',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_json_partitioned/',
     partitioned_by = ARRAY['key1'])
AS select name1, address1, comment1, key1
FROM table1;
```

Example Exemple : création de tables partitionnées et compartimentées

L'exemple suivant illustre une requête `CREATE TABLE AS SELECT` qui utilise à la fois le partitionnement et la mise en compartiments pour stocker les résultats de requête dans Simple Storage Service (Amazon S3). Les résultats de la table sont partitionnés et mis en compartiments à l'aide de différentes colonnes. Athena prend en charge un maximum de 100 combinaisons uniques de compartiments et de partitions. Par exemple, si vous créez une table avec cinq compartiments, 20 partitions avec cinq compartiments chacune sont prises en charge. Pour la syntaxe, consultez Propriétés de la table CTAS.

Pour plus d'informations sur le choix de colonnes pour la mise en compartiments, consultez Partitionnement et compartimentation dans Athena.

```
CREATE TABLE ctas_avro_bucketed
WITH (
      format = 'AVRO',
      external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_avro_bucketed/',
      partitioned_by = ARRAY['nationkey'],
      bucketed_by = ARRAY['mktsegment'],
      bucket_count = 3)
AS SELECT key1, name1, address1, phone1, acctbal, mktsegment, comment1, nationkey
FROM table1;
```

Example Exemple : Création d'une table Iceberg avec des données Parquet

L'exemple suivant crée une table Iceberg avec des fichiers de données Parquet. Les fichiers sont partitionnés par mois à l'aide de la colonne `dt` dans `table1`. L'exemple met à jour les propriétés de rétention de la table afin que 10 instantanés soient retenus par défaut sur chaque branche de la table. Les instantanés des 7 derniers jours sont également retenus. Pour plus d'informations sur les propriétés de table Iceberg dans Athena, consultez Propriétés de la table.

```
CREATE TABLE ctas_iceberg_parquet
WITH (table_type = 'ICEBERG',
      format = 'PARQUET',
      location = 's3://DOC-EXAMPLE-BUCKET/ctas_iceberg_parquet/',
      is_external = false,
      partitioning = ARRAY['month(dt)'],
      vacuum_min_snapshots_to_keep = 10,
      vacuum_max_snapshot_age_seconds = 604800
   )
AS SELECT key1, name1, dt FROM table1;
```

Example Exemple : Création d'une table Iceberg avec des données Avro

L'exemple suivant crée une table Iceberg avec des fichiers de données Avro partitionnés par `key1`.

```
CREATE TABLE ctas_iceberg_avro
WITH ( format = 'AVRO',
       location = 's3://DOC-EXAMPLE-BUCKET/ctas_iceberg_avro/',
       is_external = false,
       table_type = 'ICEBERG',
       partitioning = ARRAY['key1'])
AS SELECT key1, name1, date FROM table1;
```

Utilisation de CTAS et INSERT INTO pour ETL et l'analyse des données

Vous pouvez utiliser les instructions Create Table as Select (CTAS) et INSERT INTO dans Athena pour extraire, transformer et charger (ETL, extract-transform-load) des données dans Simple Storage Service (Amazon S3) pour leur traitement. Cette rubrique montre comment utiliser ces instructions pour partitionner et convertir un jeu de données au format de données en colonnes, afin de l'optimiser pour l'analyse des données.

Les instructions CTAS utilisent des requêtes SELECT standard pour créer de nouvelles tables. Vous pouvez utiliser une instruction CTAS pour créer un sous-ensemble de vos données à analyser. Dans

une instruction CTAS, vous pouvez partitionner les données, spécifier la compression et convertir les données dans un format en colonnes comme Apache Parquet ou Apache ORC. Lorsque vous exécutez la requête CTAS, les tables et partitions qu'elle crée sont automatiquement ajoutées au AWS Glue Data Catalog. Ainsi, les nouvelles tables et partitions créées sont immédiatement disponibles pour les requêtes suivantes.

Les instructions INSERT INTO insèrent de nouvelles lignes dans une table de destination en fonction d'une instruction de requête SELECT exécutée sur une table source. Vous pouvez utiliser les instructions INSERT INTO pour transformer et charger les données de table source au format CSV en données de table de destination à l'aide de toutes les transformations prises en charge par CTAS.

Présentation

Dans Athena, utilisez une instruction CTAS pour effectuer une conversion initiale par lots des données. Utilisez ensuite plusieurs instructions INSERT INTO pour effectuer des mises à jour incrémentielles de la table créée par l'instruction CTAS.

Étapes

- Étape 1 : créez une table basée sur le jeu de données d'origine
- Étape 2 : utiliser CTAS pour partitionner, convertir et compresser les données
- Étape 3 : utilisez INSERT INTO pour ajouter des données
- Étape 4 : mesurez les différences de performance et de coûts

Étape 1 : créez une table basée sur le jeu de données d'origine

L'exemple de cette rubrique utilise un sous-ensemble Simple Storage Service (Amazon S3) lisible du jeu de données NOAA Global Historical Climatology Network Daily (GHCN-D) accessible au public. Les données sur Simple Storage Service (Amazon S3) possèdent les caractéristiques suivantes.

```
Location: s3://aws-bigdata-blog/artifacts/athena-ctas-insert-into-blog/
Total objects: 41727
Size of CSV dataset: 11.3 GB
Region: us-east-1
```

Les données d'origine sont stockées dans Simple Storage Service (Amazon S3) sans partitions. Les données sont au format CSV dans des fichiers comme ceux ci-dessous.

```
2019-10-31 13:06:57  413.1 KiB artifacts/athena-ctas-insert-into-blog/2010.csv0000
2019-10-31 13:06:57  412.0 KiB artifacts/athena-ctas-insert-into-blog/2010.csv0001
2019-10-31 13:06:57   34.4 KiB artifacts/athena-ctas-insert-into-blog/2010.csv0002
2019-10-31 13:06:57  412.2 KiB artifacts/athena-ctas-insert-into-blog/2010.csv0100
2019-10-31 13:06:57  412.7 KiB artifacts/athena-ctas-insert-into-blog/2010.csv0101
```

Les tailles de fichier dans cet échantillon sont relativement petites. En les fusionnant dans des fichiers plus grands, vous pouvez réduire le nombre total de fichiers, ce qui permet d'augmenter les performances des requêtes. Vous pouvez utiliser les instructions CTAS et INSERT INTO pour améliorer les performances des requêtes.

Pour créer une base de données et une table à partir de l'exemple de jeu de données

1. Dans la console Athena, choisissez l'est des États-Unis (Virginie du Nord). Région AWS Assurez-vous d'exécuter toutes les requêtes dans ce didacticiel dans `us-east-1`.
2. Dans l'éditeur de requête Athena, exécutez la commande CREATE DATABASE (CRÉER UNE BASE DE DONNÉES) pour créer une base de données.

   ```
   CREATE DATABASE blogdb
   ```

3. Exécutez l'instruction suivante pour créer une table.

   ```
   CREATE EXTERNAL TABLE `blogdb`.`original_csv` (
     `id` string,
     `date` string,
     `element` string,
     `datavalue` bigint,
     `mflag` string,
     `qflag` string,
     `sflag` string,
     `obstime` bigint)
   ROW FORMAT DELIMITED
     FIELDS TERMINATED BY ','
   STORED AS INPUTFORMAT
     'org.apache.hadoop.mapred.TextInputFormat'
   OUTPUTFORMAT
     'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
   LOCATION
     's3://aws-bigdata-blog/artifacts/athena-ctas-insert-into-blog/'
   ```

Étape 2 : utiliser CTAS pour partitionner, convertir et compresser les données

Après avoir créé une table, vous pouvez utiliser une seule instruction CTAS pour convertir les données au format Parquet avec compression Snappy et pour partitionner les données par année.

Le tableau que vous avez créé à l'Étape 1 comporte un champ `date` dont la date est au format YYYYMMDD (par exemple, `20100104`). Étant donné que la nouvelle table sera partitionnée par `year`, l'exemple d'instruction de la procédure suivante utilise la fonction Presto `substr("date",1,4)` pour extraire la valeur `year` du champ `date`.

Pour convertir les données au format parquet avec compression snappy, partitionnement par année

- Exécutez l'instruction CTAS suivante, en remplaçant *votre-compartiment* par l'emplacement de votre compartiment Simple Storage Service (Amazon S3).

```
CREATE table new_parquet
WITH (format='PARQUET',
parquet_compression='SNAPPY',
partitioned_by=array['year'],
external_location = 's3://DOC-EXAMPLE-BUCKET/optimized-data/')
AS
SELECT id,
         date,
         element,
         datavalue,
         mflag,
         qflag,
         sflag,
         obstime,
         substr("date",1,4) AS year
FROM original_csv
WHERE cast(substr("date",1,4) AS bigint) >= 2015
        AND cast(substr("date",1,4) AS bigint) <= 2019
```

> **Note**
>
> Dans cet exemple, la table que vous créez inclut uniquement les données de 2015 à 2019. À l'étape 3, vous ajoutez de nouvelles données à ce tableau à l'aide de la commande INSERT INTO.

Une fois la requête terminée, utilisez la procédure suivante pour vérifier le résultat à l'emplacement Simple Storage Service (Amazon S3) spécifié dans l'instruction CTAS.

Pour voir les partitions et les fichiers Parquet créés par l'instruction CTAS

1. Pour afficher les partitions créées, exécutez la AWS CLI commande suivante. Assurez-vous d'inclure la barre oblique finale (/).

   ```
   aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/
   ```

   Le résultat affiche les partitions.

   ```
         PRE year=2015/
         PRE year=2016/
         PRE year=2017/
         PRE year=2018/
         PRE year=2019/
   ```

2. Pour afficher les fichiers Parquet, exécutez la commande suivante. Notez que l'option | head -5 qui limite la sortie aux cinq premiers résultats, n'est pas disponible sous Windows.

   ```
   aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/ --recursive --human-readable |
    head -5
   ```

   La sortie se présente comme suit :

   ```
   2019-10-31 14:51:05    7.3 MiB optimized-data/
   year=2015/20191031_215021_00001_3f42d_1be48df2-3154-438b-b61d-8fb23809679d
   2019-10-31 14:51:05    7.0 MiB optimized-data/
   year=2015/20191031_215021_00001_3f42d_2a57f4e2-ffa0-4be3-9c3f-28b16d86ed5a
   2019-10-31 14:51:05    9.9 MiB optimized-data/
   year=2015/20191031_215021_00001_3f42d_34381db1-00ca-4092-bd65-ab04e06dc799
   2019-10-31 14:51:05    7.5 MiB optimized-data/
   year=2015/20191031_215021_00001_3f42d_354a2bc1-345f-4996-9073-096cb863308d
   2019-10-31 14:51:05    6.9 MiB optimized-data/
   year=2015/20191031_215021_00001_3f42d_42da4cfd-6e21-40a1-8152-0b902da385a1
   ```

Étape 3 : utilisez INSERT INTO pour ajouter des données

À l'Étape 2, vous avez utilisé CTAS pour créer une table avec des partitions pour les années 2015 à 2019. Toutefois, le jeu de données d'origine contient également des données pour les années 2010 à 2014. Maintenant, vous ajoutez ces données à l'aide d'une instruction INSERT INTO.

Pour ajouter des données à la table à l'aide d'une ou plusieurs instructions INSERT INTO

1. Exécutez la commande INSERT INTO suivante, en spécifiant les années avant 2015 dans la clause OÙ.

```
INSERT INTO new_parquet
SELECT id,
         date,
         element,
         datavalue,
         mflag,
         qflag,
         sflag,
         obstime,
         substr("date",1,4) AS year
FROM original_csv
WHERE cast(substr("date",1,4) AS bigint) < 2015
```

2. Exécutez à nouveau la commande `aws s3 ls` en utilisant la syntaxe suivante.

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/
```

Le résultat affiche les nouvelles partitions.

```
      PRE year=2010/
      PRE year=2011/
      PRE year=2012/
      PRE year=2013/
      PRE year=2014/
      PRE year=2015/
      PRE year=2016/
      PRE year=2017/
      PRE year=2018/
      PRE year=2019/
```

3. Pour voir la réduction de la taille du jeu de données obtenue à l'aide de la compression et du stockage en colonnes au format Parquet, exécutez la commande suivante.

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/ --recursive --human-readable --
summarize
```

Les résultats suivants montrent que la taille du jeu de données après Parquet avec compression Snappy est de 1,2 Go.

```
...
2020-01-22 18:12:02 2.8 MiB optimized-data/
year=2019/20200122_181132_00003_nja5r_f0182e6c-38f4-4245-afa2-9f5bfa8d6d8f
2020-01-22 18:11:59 3.7 MiB optimized-data/
year=2019/20200122_181132_00003_nja5r_fd9906b7-06cf-4055-a05b-f050e139946e
Total Objects: 300
     Total Size: 1.2 GiB
```

4. Si d'autres données CSV sont ajoutées à la table d'origine, vous pouvez ajouter ces données à la table Parquet à l'aide d'instructions INSERT INTO. Par exemple, si vous possédez de nouvelles données pour l'année 2020, vous pouvez exécuter l'instruction INSERT INTO suivante. L'instruction ajoute les données et la partition correspondante à la table `new_parquet`.

```
INSERT INTO new_parquet
SELECT id,
         date,
         element,
         datavalue,
         mflag,
         qflag,
         sflag,
         obstime,
         substr("date",1,4) AS year
FROM original_csv
WHERE cast(substr("date",1,4) AS bigint) = 2020
```

> **Note**
>
> L'instruction INSERT INTO prend en charge l'écriture de 100 partitions au maximum dans la table de destination. Toutefois, pour ajouter plus de 100 partitions, vous pouvez

exécuter plusieurs instructions INSERT INTO. Pour plus d'informations, consultez Utilisation de CTAS et de INSERT INTO pour contourner la limite de 100 partitions.

Étape 4 : mesurez les différences de performance et de coûts

Après avoir transformé les données, vous pouvez mesurer les gains en termes de performance et d'économies en exécutant les mêmes requêtes sur les nouvelles tables et les anciennes, puis en comparant les résultats.

> **Note**
>
> Pour obtenir des informations sur les coûts par requête Athena, consultez la rubrique Tarification Amazon Athena.

Mesurer les gains de performance et les différences de coûts

1. Exécutez la requête suivante sur la table d'origine. La requête recherche le nombre d'ID distincts pour chaque valeur de l'année.

```
SELECT substr("date",1,4) as year,
       COUNT(DISTINCT id)
FROM original_csv
GROUP BY 1 ORDER BY 1 DESC
```

2. Notez la durée d'exécution de la requête et la quantité de données analysées.
3. Exécutez la même requête sur la nouvelle table, en notant le temps d'exécution de la requête et la quantité de données analysées.

```
SELECT year,
  COUNT(DISTINCT id)
FROM new_parquet
GROUP BY 1 ORDER BY 1 DESC
```

4. Comparez les résultats et calculez la différence de performance et de coût. Les exemples de résultats suivants montrent que la requête de test sur la nouvelle table a été plus rapide et moins chère que la requête sur l'ancienne table.

Tableau	Environnement d’exécution	Données analysées
Original	16,88 secondes	11,35 Go
New	3,79 secondes	428,05 Mo

5. Exécutez l'exemple de requête suivant sur la table d'origine. La requête calcule la température maximale moyenne (Celsius), la température minimale moyenne (Celsius) et la pluviométrie moyenne (mm) pour la Terre en 2018.

```
SELECT element, round(avg(CAST(datavalue AS real)/10),2) AS value
FROM original_csv
WHERE element IN ('TMIN', 'TMAX', 'PRCP') AND substr("date",1,4) = '2018'
GROUP BY 1
```

6. Notez la durée d'exécution de la requête et la quantité de données analysées.
7. Exécutez la même requête sur la nouvelle table, en notant le temps d'exécution de la requête et la quantité de données analysées.

```
SELECT element, round(avg(CAST(datavalue AS real)/10),2) AS value
FROM new_parquet
WHERE element IN ('TMIN', 'TMAX', 'PRCP') and year = '2018'
GROUP BY 1
```

8. Comparez les résultats et calculez la différence de performance et de coût. Les exemples de résultats suivants montrent que la requête de test sur la nouvelle table a été plus rapide et moins chère que la requête sur l'ancienne table.

Tableau	Environnement d’exécution	Données analysées
Original	18,65 secondes	11,35 Go
New	1,92 secondes	68 Mo

Récapitulatif

Cette rubrique vous a montré comment effectuer des opérations ETL à l'aide des instructions CTAS et INSERT INTO dans Athena. Vous avez effectué le premier ensemble de transformations à l'aide

d'une instruction CTAS qui a converti les données au format Parquet avec compression Snappy. L'instruction CTAS a également converti le jeu de données non partitionné en partitionné. Vous avez ainsi réduit sa taille et les coûts d'exécution des requêtes. Lorsque de nouvelles données sont disponibles, vous pouvez utiliser une instruction INSERT INTO pour transformer et charger les données dans la table que vous avez créée avec l'instruction CTAS.

Utilisation de CTAS et de INSERT INTO pour contourner la limite de 100 partitions

Athena a une limite de 100 partitions par requête `CREATE TABLE AS SELECT` (CTAS). De même, vous pouvez ajouter un maximum de 100 partitions à une table de destination avec une instruction INSERT INTO.

Si vous dépassez cette limite, le message d'erreur HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets (Dépassement de la limite de 100 rédacteurs ouverts pour les partitions/compartiments) peut s'afficher. Pour contourner ces limitations, vous pouvez utiliser une instruction CTAS et une série d'instructions `INSERT INTO` qui créent ou insèrent jusqu'à 100 partitions chacune.

L'exemple présenté dans cette rubrique utilise une base de données appelée `tpch100` dont les données se trouvent dans l'emplacement du compartiment Amazon S3 s3://DOC-EXAMPLE-BUCKET/.

Pour utiliser CTAS et INSERT INTO pour créer une table de plus de 100 partitions

1. Utilisez une instruction `CREATE EXTERNAL TABLE` pour créer une table partitionnée sur le champ souhaité.

   L'instruction exemple suivante partitionne les données selon la colonne `l_shipdate`. La table a 2 525 partitions.

```
CREATE EXTERNAL TABLE `tpch100.lineitem_parq_partitioned`(
  `l_orderkey` int,
  `l_partkey` int,
  `l_suppkey` int,
  `l_linenumber` int,
  `l_quantity` double,
  `l_extendedprice` double,
  `l_discount` double,
  `l_tax` double,
```

```
  `l_returnflag` string,
  `l_linestatus` string,
  `l_commitdate` string,
  `l_receiptdate` string,
  `l_shipinstruct` string,
  `l_comment` string)
PARTITIONED BY (
  `l_shipdate` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe' STORED AS
 INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat' OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat' LOCATION
 's3://DOC-EXAMPLE-BUCKET/lineitem/'
```

2. Exécutez une commande `SHOW PARTITIONS <table_name>` comme la suivante pour répertorier les partitions.

```
SHOW PARTITIONS lineitem_parq_partitioned
```

Des résultats d'échantillon partiels sont présentés ci-dessous.

```
/*
l_shipdate=1992-01-02
l_shipdate=1992-01-03
l_shipdate=1992-01-04
l_shipdate=1992-01-05
l_shipdate=1992-01-06

...

l_shipdate=1998-11-24
l_shipdate=1998-11-25
l_shipdate=1998-11-26
l_shipdate=1998-11-27
l_shipdate=1998-11-28
l_shipdate=1998-11-29
l_shipdate=1998-11-30
l_shipdate=1998-12-01
*/
```

3. Exécutez une requête CTAS pour créer une table partitionnée.

L'exemple suivant montre comment créer une table appelée `my_lineitem_parq_partitioned` et utiliser la clause `WHERE` pour restreindre la `DATE` à une date antérieure à `1992-02-01`. Étant donné que l'exemple de jeu de données commence en janvier 1992, seules les partitions de janvier 1992 sont créées.

```
CREATE table my_lineitem_parq_partitioned
WITH (partitioned_by = ARRAY['l_shipdate']) AS
SELECT l_orderkey,
         l_partkey,
         l_suppkey,
         l_linenumber,
         l_quantity,
         l_extendedprice,
         l_discount,
         l_tax,
         l_returnflag,
         l_linestatus,
         l_commitdate,
         l_receiptdate,
         l_shipinstruct,
         l_comment,
         l_shipdate
FROM tpch100.lineitem_parq_partitioned
WHERE cast(l_shipdate as timestamp) < DATE ('1992-02-01');
```

4. Exécutez la commande `SHOW PARTITIONS` pour vérifier que la table contient les partitions souhaitées.

```
SHOW PARTITIONS my_lineitem_parq_partitioned;
```

Les partitions de l'exemple datent de janvier 1992.

```
/*
l_shipdate=1992-01-02
l_shipdate=1992-01-03
l_shipdate=1992-01-04
l_shipdate=1992-01-05
l_shipdate=1992-01-06
l_shipdate=1992-01-07
l_shipdate=1992-01-08
l_shipdate=1992-01-09
```

```
l_shipdate=1992-01-10
l_shipdate=1992-01-11
l_shipdate=1992-01-12
l_shipdate=1992-01-13
l_shipdate=1992-01-14
l_shipdate=1992-01-15
l_shipdate=1992-01-16
l_shipdate=1992-01-17
l_shipdate=1992-01-18
l_shipdate=1992-01-19
l_shipdate=1992-01-20
l_shipdate=1992-01-21
l_shipdate=1992-01-22
l_shipdate=1992-01-23
l_shipdate=1992-01-24
l_shipdate=1992-01-25
l_shipdate=1992-01-26
l_shipdate=1992-01-27
l_shipdate=1992-01-28
l_shipdate=1992-01-29
l_shipdate=1992-01-30
l_shipdate=1992-01-31
*/
```

5. Utilisez une instruction `INSERT INTO` pour ajouter des partitions à la table.

   L'exemple suivant montre comment ajouter des partitions pour les dates du mois de février 1992.

```
INSERT INTO my_lineitem_parq_partitioned
SELECT l_orderkey,
         l_partkey,
         l_suppkey,
         l_linenumber,
         l_quantity,
         l_extendedprice,
         l_discount,
         l_tax,
         l_returnflag,
         l_linestatus,
         l_commitdate,
         l_receiptdate,
         l_shipinstruct,
         l_comment,
         l_shipdate
```

```
FROM tpch100.lineitem_parq_partitioned
WHERE cast(l_shipdate as timestamp) >= DATE ('1992-02-01')
AND cast(l_shipdate as timestamp) < DATE ('1992-03-01');
```

6. Exécutez à nouveau `SHOW PARTITIONS`.

```
SHOW PARTITIONS my_lineitem_parq_partitioned;
```

La table d'exemple comporte maintenant des partitions de janvier et de février 1992.

```
/*
l_shipdate=1992-01-02
l_shipdate=1992-01-03
l_shipdate=1992-01-04
l_shipdate=1992-01-05
l_shipdate=1992-01-06

...

l_shipdate=1992-02-20
l_shipdate=1992-02-21
l_shipdate=1992-02-22
l_shipdate=1992-02-23
l_shipdate=1992-02-24
l_shipdate=1992-02-25
l_shipdate=1992-02-26
l_shipdate=1992-02-27
l_shipdate=1992-02-28
l_shipdate=1992-02-29
*/
```

7. Continuez à utiliser des instructions `INSERT INTO` qui ne lisent et n'ajoutent pas plus de 100 partitions chacune. Continuez jusqu'à atteindre le nombre de partitions dont vous avez besoin.

> **Important**
>
> Lorsque vous définissez la condition `WHERE`, assurez-vous que les requêtes ne se chevauchent pas. Sinon, certaines partitions peuvent comprendre des données dupliquées.

Référence SerDe

Athena prend en charge plusieurs bibliothèques SerDe pour l'analyse des données à partir de différents formats de données, tels que CSV, JSON, Parquet et ORC. Athena ne prend pas en charge les SerDe personnalisés.

Rubriques

À l'aide d'un SerDe

Un SerDe (sérialiseur/désérialiseur) est un moyen par lequel Athena interagit avec des données dans différents formats.

C'est ce que SerDe vous spécifiez, et non le DDL, qui définit le schéma de table. En d'autres termes, ils SerDe peuvent remplacer la configuration DDL que vous spécifiez dans Athena lorsque vous créez votre table.

Pour utiliser un SerDe dans les requêtes

Pour utiliser un SerDe lors de la création d'une table dans Athena, appliquez l'une des méthodes suivantes :

- Spécifiez `ROW FORMAT DELIMITED`, puis utilisez des instructions DDL pour spécifier des délimiteurs de champs, comme dans l'exemple suivant. Lorsque vous le spécifiez`ROW FORMAT DELIMITED`, Athena utilise le LazySimpleSerDe par défaut.

  ```
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  ESCAPED BY '\\'
  COLLECTION ITEMS TERMINATED BY '|'
  MAP KEYS TERMINATED BY ':'
  ```

  Pour des exemples de `ROW FORMAT DELIMITED`, consultez les rubriques suivantes :

  LazySimpleSerDe pour les fichiers CSV, TSV et délimités de manière personnalisée

  Interrogation des journaux Amazon CloudFront

Interrogation des journaux Amazon EMR

Interrogation des journaux de flux Amazon VPC

Utilisation de CTAS et INSERT INTO pour ETL et l'analyse des données

- `ROW FORMAT SERDE`À utiliser pour spécifier explicitement le type SerDe qu'Athéna doit utiliser lorsqu'elle lit et écrit des données dans la table. L'exemple suivant spécifie le LazySimpleSerDe. Pour spécifier les délimiteurs, utilisez `WITH SERDEPROPERTIES`. Les propriétés spécifiées par `WITH SERDEPROPERTIES` correspondent aux instructions séparées (comme `FIELDS TERMINATED BY`) dans l'exemple `ROW FORMAT DELIMITED`.

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
'serialization.format' = ',',
'field.delim' = ',',
'collection.delim' = '|',
'mapkey.delim' = ':',
'escape.delim' = '\\'
)
```

Pour des exemples de `ROW FORMAT SERDE`, consultez les rubriques suivantes :

SerDe Avro

Grok SerDe

SerDe bibliothèques JSON

SerDe OpenCSV pour le traitement des fichiers CSV

Régex SerDe

SerDe et formats de données pris en charge

Athena prend en charge la création de tables et l'interrogation de données à partir de fichiers aux formats CSV, TSV, avec séparateur personnalisé et JSON ; de données de formats associés à Hadoop : ORC, Apache Avro et Parquet ; de fichiers journaux de Logstash, de journaux AWS CloudTrail et de journaux Apache WebServer.

> **Note**
>
> Les formats répertoriés dans cette section sont utilisés par Athena pour la lecture des données. Pour plus d'informations sur les formats utilisés par Athena pour écrire des données lors de l'exécution des requêtes CTAS, voir Création d'une table à partir des résultats des requêtes (CTAS).

Pour créer des tables et interroger les données de fichiers sous ces formats dans Athena, spécifiez une classe sérialiseur-désérialiseur (SerDe) afin qu'Athena connaisse le format utilisé et sache comment analyser les données.

Cette table répertorie les formats de données pris en charge dans Athena et leurs bibliothèques SerDe correspondantes.

Un SerDe est une bibliothèque personnalisée qui indique au catalogue de données utilisé par Athena comment traiter les données. Vous spécifiez un type SerDe en l'énumérant explicitement dans la partie `ROW FORMAT` de votre instruction `CREATE TABLE` dans Athena. Dans certains cas, vous pouvez omettre le nom SerDe, car Athena utilise certains types SerDe par défaut pour certains types de formats de données.

Formats de données et SerDe pris en charge

Format de données	Description	Types SerDe pris en charge dans Athena
Amazon Ion	Amazon Ion est un format de données abondamment typé et auto-descriptif qui est un superensemble de JSON, développé et ouvert par Amazon.	Utilisez SerDe Amazon Ion Hive.
Apache Avro	Format d'enregistrement des données dans Hadoop qui utilise des schémas basés sur JSON pour les valeurs d'enregistrement.	Utilisez SerDe Avro.

Format de données	Description	Types SerDe pris en charge dans Athena
Apache Parquet	Format pour le stockage en colonnes des données dans Hadoop.	Utilisez le type Parquet SerDe et la compression SNAPPY.
Fichiers journaux Apache WebServer	Format pour le stockage des fichiers journaux dans Apache WebServer.	Utilisez le type Grok SerDe ou Régex SerDe.
Journaux CloudTrail	Format pour le stockage des fichiers journaux dans CloudTrail.	• Utilisez Hive JSON SerDe. Pour de plus amples informations, veuillez consulter Journaux d'interrogation AWS CloudTrail.
CSV (valeurs séparées par des virgules)	Pour les données au format CSV, chaque ligne représente un enregistrement de données, et chaque enregistrement se compose d'un ou de plusieurs champs, séparés par des virgules.	• Utilisez le type LazySimpleSerDe pour les fichiers CSV, TSV et délimités de manière personnalisée si vos données n'incluent pas de valeurs entre guillemets ou si elles utilisent le format `java.sql.Timestamp` . • Utilisez le type SerDe OpenCSV pour le traitement des fichiers CSV lorsque vos données comprennent des guillemets dans les valeurs ou utilisent le format numérique UNIX pour `TIMESTAMP` (par exemplee, `1564610311`).

Format de données	Description	Types SerDe pris en charge dans Athena
Séparateur personnalisé	Pour les données qui se trouvent dans ce format, chaque ligne représente un enregistrement de données. Les enregistrements sont séparés par des délimiteurs personnalisés.	Utilisez le type LazySimpleSerDe pour les fichiers CSV, TSV et délimités de manière personnalisée et spécifiez un séparateur à caractère unique personnalisé.
JSON (JavaScript Object Notation)	Pour les données JSON, chaque ligne représente un enregistrement de données, et chaque enregistrement se compose de paires attribut-valeur et de tableaux, séparés par des virgules.	• Utilisez Hive JSON SerDe. • Utilisez OpenX JSON SerDe.
Journaux Logstash	Format pour le stockage des fichiers journaux dans Logstash.	Utilisez Grok SerDe.
ORC (Optimized Row Columnar)	Format pour le stockage en colonnes optimisé des données Hive.	Utilisez le type ORC SerDe et la compression ZLIB.
TSV (valeurs séparées par des tabulations)	Pour les données au format TSV, chaque ligne représente un enregistrement de données, et chaque enregistrement se compose d'un ou de plusieurs champs, séparés par des tabulations.	Utilisez le type LazySimpleSerDe pour les fichiers CSV, TSV et délimités de manière personnalisée et spécifiez le caractère séparateur sous la forme `FIELDS TERMINATED BY '\t'`.

Rubriques

- SerDe Amazon Ion Hive

- SerDe Avro
- Grok SerDe
- SerDe bibliothèques JSON
- LazySimpleSerDe pour les fichiers CSV, TSV et délimités de manière personnalisée
- SerDe OpenCSV pour le traitement des fichiers CSV
- ORC SerDe
- Parquet SerDe
- Régex SerDe

SerDe Amazon Ion Hive

Vous pouvez utiliser le SerDe Amazon Ion Hive pour interroger des données stockées au format Amazon Ion. Amazon Ion est un format de données open source riche et auto-descriptif. Le format Amazon Ion est utilisé par des services tels que Amazon Quantum Ledger Database(Amazon QLDB) et dans le langage de requête SQL open source PartiQL.

Amazon Ion possède des formats binaires et texte interchangeables. Cette fonction combine la facilité d'utilisation du texte avec l'efficacité du codage binaire.

Pour interroger les données Amazon Ion depuis Athena, vous pouvez utiliser le SerDe Amazon Ion Hive, qui sérialise et désérialise les données Amazon Ion. La désérialisation vous permet d'exécuter des requêtes sur les données Amazon Ion ou de les lire pour écrire dans un format différent comme Parquet ou ORC. La sérialisation vous permet de générer des données au format Amazon Ion en utilisant des requêtes `CREATE TABLE AS SELECT` (CTAS) ou `INSERT INTO` pour copier des données à partir de tables existantes.

> **Note**
>
> Comme Amazon Ion est un sur-ensemble de JSON, vous pouvez utiliser le SerDe Amazon Ion Hive pour interroger des jeux de données JSON non Amazon Ion. Contrairement à d'autres bibliothèques SerDe JSON, le SerDe Amazon Ion ne s'attend pas à ce que chaque ligne de données soit sur une seule ligne. Cette fonction est utile si vous souhaitez interroger des jeux de données JSON au format « pretty print » ou si vous souhaitez diviser les champs d'une ligne avec des caractères de saut de ligne.

Pour obtenir des informations supplémentaires et des exemples d'interrogation d'Amazon Ion avec Athena, consultez Analyser des ensembles de données Amazon Ion à l'aide d'Amazon Athena.

Nom du SerDe

- com.amazon.ionhiveserde.IonHiveSerDe

Considérations et restrictions

- Champs dupliqués – Les structures Amazon Ion sont commandées et prennent en charge les champs dupliqués, tandis que ce n'est pas le cas pour les structures `STRUCT<>` et `MAP<>`. Ainsi, lorsque vous désérialisez un champ dupliqué à partir d'une structure Amazon Ion, une seule valeur est choisie de façon non déterministe et les autres sont ignorées.
- Tableaux de symboles externes non pris en charge – Actuellement, Athena ne prend pas en charge les tables de symboles externes, ni les propriétés SerDe d'Amazon Ion Hive suivantes :
  - `ion.catalog.class`
  - `ion.catalog.file`
  - `ion.catalog.url`
  - `ion.symbol_table_imports`
- Extensions de fichier – Amazon Ion utilise des extensions de fichiers pour déterminer quel codec de compression utiliser pour désérialiser les fichiers Amazon Ion. Par conséquent, les fichiers compressés doivent avoir l'extension de fichier correspondant à l'algorithme de compression utilisé. Par exemple, si ZSTD est utilisé, les fichiers correspondants doivent porter l'extension `.zst`.
- Données homogènes – Amazon Ion n'a aucune restriction sur les types de données qui peuvent être utilisés pour des valeurs dans des champs particuliers. Par exemple, deux documents Amazon Ion différents peuvent comporter un champ portant le même nom et possédant des types de données différents. Toutefois, comme Hive utilise un schéma, toutes les valeurs que vous extrayez dans une seule colonne Hive doivent avoir le même type de données.
- Restrictions relatives aux types de clés – Lorsque vous sérialisez des données d'un autre format dans Amazon Ion, assurez-vous que le type de clé de mappage est l'un des suivants : `STRING`, `VARCHAR`, ou `CHAR`. Bien que Hive vous permette d'utiliser n'importe quel type de données primitif comme clé de mappage, Amazon Ion symbols (symboles Amazon Ion) doit être un type de chaîne.
- Type union – Athena ne prend pas actuellement en charge le type union de Hive.
- Type de données double : Amazon Ion ne prend actuellement pas en charge le type de données `double`.

Rubriques

Utilisation de CREATE TABLE pour créer des tables Amazon Ion

Pour créer une table dans Athena à partir de données stockées au format Amazon Ion, vous pouvez utiliser l'une des techniques suivantes dans une instruction CREATE TABLE :

- Spécifiez `STORED AS ION`. Dans le cadre de cette utilisation, il n'est pas nécessaire de spécifier l'Amazon Ion Hive de SerDe manière explicite. Ce choix est l'option la plus simple.
- Spécifiez les chemins d'accès de classe Amazon Ion dans les champs `ROW FORMAT SERDE`, `INPUTFORMAT`, et `OUTPUTFORMAT`.

Vous pouvez également utiliser des instructions `CREATE TABLE AS SELECT` (CTAS) pour créer des tables Amazon Ion dans Athena. Pour plus d'informations, veuillez consulter Utilisation de CTAS et de INSERT INTO pour créer des tables Amazon Ion.

Spécification de STORED AS ION

L'exemple d'instruction `CREATE TABLE` suivant utilise la `STORED AS ION` avant la clause `LOCATION` pour créer une table basée sur des données de vol au format Amazon Ion. La clause `LOCATION` indique le compartiment ou le dossier où se trouvent les fichiers d'entrée au format Ion. Tous les fichiers se trouvant à l'emplacement spécifié sont analysés.

```
CREATE EXTERNAL TABLE flights_ion (
    yr INT,
    quarter INT,
    month INT,
    dayofmonth INT,
    dayofweek INT,
    flightdate STRING,
    uniquecarrier STRING,
    airlineid INT,
)
STORED AS ION
```

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
```

Spécification des chemins de classe Amazon Ion

Au lieu d'utiliser la syntaxe `STORED AS ION`, vous pouvez spécifier explicitement les valeurs de chemin d'accès de classe Ion pour les clauses `ROW FORMAT SERDE`, `INPUTFORMAT`, et `OUTPUTFORMAT` comme suit.

Paramètre	Chemin de classe Ion
`ROW FORMAT SERDE`	`'com.amazon.ionhiveserde.IonHiveSerDe'`
`STORED AS INPUTFORMAT`	`'com.amazon.ionhiveserde.formats.IonInputFormat'`
`OUTPUTFORMAT`	`'com.amazon.ionhiveserde.formats.IonOutputFormat'`

La requête DDL suivante utilise cette technique pour créer la même table externe que dans l'exemple précédent.

```
CREATE EXTERNAL TABLE flights_ion (
    yr INT,
    quarter INT,
    month INT,
    dayofmonth INT,
    dayofweek INT,
    flightdate STRING,
    uniquecarrier STRING,
    airlineid INT,
)
ROW FORMAT SERDE
 'com.amazon.ionhiveserde.IonHiveSerDe'
STORED AS INPUTFORMAT
 'com.amazon.ionhiveserde.formats.IonInputFormat'
OUTPUTFORMAT
 'com.amazon.ionhiveserde.formats.IonOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
```

Pour plus d'informations sur les SerDe propriétés des `CREATE TABLE` déclarations dans Athéna, consultez. Utilisation des propriétés SerDe Amazon Ion

Utilisation de CTAS et de INSERT INTO pour créer des tables Amazon Ion

Vous pouvez utiliser les instructions `CREATE TABLE AS SELECT` (CTAS) et `INSERT INTO` pour copier ou insérer des données d'une table dans une nouvelle table au format Amazon Ion dans Athena.

Dans une requête CTAS, spécifiez `format='ION'` dans la clause `WITH`, comme dans l'exemple suivant.

```
CREATE TABLE new_table
WITH (format='ION')
AS SELECT * from existing_table
```

Par défaut, Athena sérialise les résultats Amazon Ion dans un format binaire Ion, mais vous pouvez également utiliser le format texte. Pour utiliser un format texte, spécifiez `ion_encoding = 'TEXT'` dans la clause `WITH` CTAS, comme dans l'exemple suivant.

```
CREATE TABLE new_table
WITH (format='ION', ion_encoding = 'TEXT')
AS SELECT * from existing_table
```

Pour plus d'informations sur les propriétés spécifiques à Amazon Ion dans la clause `WITH` CTAS, consultez la section suivante.

Propriétés Amazon Ion de la clause WITH CTAS

Dans une requête CTAS, vous pouvez utiliser la clause `WITH` pour spécifier le format Amazon Ion et éventuellement spécifier l'algorithme de codage Amazon Ion et/ou de compression d'écriture à utiliser.

format

Vous pouvez spécifier le mot-clé `ION` comme option de format dans la clause `WITH` d'une requête CTAS. Lorsque vous le faites, la table que vous créez utilise le format que vous spécifiez pour `IonInputFormat` pour les lectures, et il sérialise les données dans le format que vous spécifiez pour `IonOutputFormat`.

L'exemple suivant indique que la requête CTAS utilise le format Amazon Ion.

```
WITH (format='ION')
```

ion_encoding

Facultatif

Par défaut : `BINARY`

Valeurs: `BINARY`, `TEXT`

Spécifie si les données sont sérialisées au format binaire Amazon Ion ou au format texte Amazon Ion. L'exemple suivant spécifie le format de texte Amazon Ion.

```
WITH (format='ION', ion_encoding='TEXT')
```

write_compression

Facultatif

Par défaut : `GZIP`

Valeurs : `GZIP`, `ZSTD`, `BZIP2`, `SNAPPY`, `NONE`

Spécifie l'algorithme de compression à utiliser pour compresser les fichiers de sortie.

L'exemple suivant indique que la requête CTAS écrit sa sortie au format Amazon Ion à l'aide de l'algorithme de compression Zstandard.

```
WITH (format='ION', write_compression = 'ZSTD')
```

Pour plus d'informations sur l'utilisation de la compression sur Athena, consultez Prise en charge de la compression Athena.

Pour plus d'informations sur les autres propriétés CTAS dans Athena, consultez Propriétés de la table CTAS.

Utilisation des propriétés SerDe Amazon Ion

Cette rubrique contient des informations sur les propriétés SerDe pour les instructions `CREATE TABLE` dans Athena. Pour de plus amples informations et des exemples d'utilisation des propriétés

Amazon Ion SerDe, consultez SerDe properties (Propriétés SerDe) dans la documentation SerDe Amazon Ion Hive sur GitHub.

Spécification des propriétés SerDe Amazon Ion

Pour spécifier les propriétés du SerDe Amazon Ion Hive dans votre instruction `CREATE TABLE`, utilisez la clause `WITH SERDEPROPERTIES`. Étant donné que `WITH SERDEPROPERTIES` est un sous-champ de la clause `ROW FORMAT SERDE`, vous devez spécifier `ROW FORMAT SERDE` et le chemin d'accès de classe SerDe Amazon Ion Hive en premier, comme le montre la syntaxe suivante.

```
...
ROW FORMAT SERDE
 'com.amazon.ionhiveserde.IonHiveSerDe'
WITH SERDEPROPERTIES (
 'property' = 'value',
 'property' = 'value',
...
)
```

Remarque : bien que la clause `ROW FORMAT SERDE` est obligatoire si vous voulez utiliser `WITH SERDEPROPERTIES`, vous pouvez utiliser `STORED AS ION` ou le plus long `INPUTFORMAT` et la syntaxe `OUTPUTFORMAT` pour spécifier le format Amazon Ion.

Propriétés SerDe Amazon Ion

Voici les propriétés SerDe Amazon Ion qui peuvent être utilisées dans les instructions `CREATE TABLE` dans Athena.

ion.codage

Facultatif

Par défaut : `BINARY`

Valeurs: `BINARY`, `TEXT`

Cette propriété déclare si les nouvelles valeurs ajoutées sont sérialisées en tant que binaire Amazon Ion ou au format texte Amazon Ion.

L'exemple de propriété SerDe suivant spécifie le format de texte Amazon Ion.

```
'ion.encoding' = 'TEXT'
```

ion.fail_on_overflow

Facultatif

Par défaut : `true`

Valeurs: `true`, `false`

Amazon Ion autorise des types numériques de taille arbitraire, tandis que Hive ne le fait pas. Par défaut, le SerDe échoue si la valeur Amazon Ion ne correspond pas à la colonne Hive, mais vous pouvez utiliser l'option de configuration `fail_on_overflow` pour laisser la valeur déborder au lieu d'échouer.

Cette propriété peut être définie au niveau de la table ou de la colonne. Pour le spécifier au niveau de la table, spécifiez `ion.fail_on_overflow` comme dans l'exemple suivant. Cela définit le comportement par défaut de toutes les colonnes.

```
'ion.fail_on_overflow' = 'true'
```

Pour contrôler une colonne spécifique, spécifiez le nom de la colonne entre `ion` et `fail_on_overflow`, délimité par des points, comme dans l'exemple suivant.

```
'ion.<column>.fail_on_overflow' = 'false'
```

ion.path_extractor.case_sensitive

Facultatif

Par défaut : `false`

Valeurs: `true`, `false`

Détermine s'il convient de traiter les noms de champs Amazon Ion comme sensibles à la casse. Quand la valeur est `false`, SerDe ignore l'analyse de casse des noms de champs Amazon Ion.

Par exemple, supposons que vous ayez un schéma de table Hive qui définit un champ `alias` en minuscules et un document Amazon Ion avec un champ `alias` et un champ `ALIAS`, comme dans l'exemple suivant.

```
-- Hive Table Schema
alias: STRING
```

```
-- Amazon Ion Document
{ 'ALIAS': 'value1'}
{ 'alias': 'value2'}
```

L'exemple suivant montre les propriétés SerDe et la table extraite qui en résulte lorsque la sensibilité à la casse est définie sur `false` :

```
-- Serde properties
'ion.alias.path_extractor' = '(alias)'
'ion.path_extractor.case_sensitive' = 'false'

--Extracted Table
| alias    |
|----------|
| "value1" |
| "value2" |
```

L'exemple suivant montre les propriétés SerDe et la table extraite qui en résulte lorsque la sensibilité à la casse est définie sur `true` :

```
-- Serde properties
'ion.alias.path_extractor' = '(alias)'
'ion.path_extractor.case_sensitive' = 'true'

--Extracted Table
| alias    |
|----------|
| "value2" |
```

Dans le second cas, la valeur `value1` pour le champ `ALIAS` est ignorée lorsque la sensibilité à la casse est définie sur `true` et l'extracteur de chemin est spécifié comme suit : `alias`.

ion.*<colonne>*.path_extractor

Facultatif

Valeur par défaut : NA

Valeurs : chaîne avec chemin de recherche

Crée un extracteur de chemin avec le chemin de recherche spécifié pour la colonne donnée. Les extracteurs de chemins mappent les champs Amazon Ion aux colonnes Hive. Si aucun extracteur

de chemin n'est spécifié, Athena crée dynamiquement des extracteurs de chemin au moment de l'exécution en fonction des noms de colonnes.

L'exemple d'extracteur de chemin suivant mappe le champ `example_ion_field` vers la colonne `example_hive_column`.

```
'ion.example_hive_column.path_extractor' = '(example_ion_field)'
```

Pour plus d'informations sur les extracteurs de chemin d'accès et les chemins de recherche, consultez Utilisation d'extracteurs de chemin.

ion.timestamp.serialization_offset

Facultatif

Par défaut : `'Z'`

Valeurs : `OFFSET`, où `OFFSET` est représenté par *<signal>*`hh:mm`. Exemples de valeurs : `01:00`, `+01:00`, `-09:30`, `Z` (UTC, identique à 00:00)

Contrairement aux horodatages Apache Hive qui n'ont pas de fuseau horaire intégré et sont stockés sous forme de décalage par rapport à l'époque UNIX, les horodatages Amazon Ion ont un décalage. Utilisez cette propriété pour spécifier le décalage lorsque vous sérialisez sur Amazon Ion.

L'exemple suivant montre comment ajouter un décalage d'une heure.

```
'ion.timestamp.serialization_offset' = '+01:00'
```

ion.serialize_null

Facultatif

Par défaut : `OMIT`

Valeurs : `OMIT`, `UNTYPED`, `TYPED`

Le SerDe Amazon Ion peut être configuré pour sérialiser ou omettre des colonnes comportant des valeurs nulles. Vous pouvez choisir d'écrire des valeurs nulles fortement typées (`TYPED`) ou des valeurs nulles non typées (`UNTYPED`). Les valeurs null fortement typées sont déterminées en fonction du mappage de type Amazon Ion vers Hive par défaut.

L'exemple suivant spécifie des valeurs nulles fortement typées.

```
'ion.serialize_null'='TYPED'
```

ion.ignore_malformed

Facultatif

Par défaut : `false`

Valeurs: `true`, `false`

Quand la valeur est `true`, ignore les entrées mal formées ou le fichier entier si SerDe n'est pas en mesure de le lire. Pour de plus amples informations, consultez la section Ignore malformed (ignorer les entrées mal formées) dans la documentation sur GitHub.

ion.*<colonne>*.serialize_as

Facultatif

Par défaut : type par défaut de la colonne.

Valeurs : chaîne contenant le type Amazon Ion

Détermine le type de données Amazon Ion dans lequel une valeur est sérialisée. Étant donné que les types Amazon Ion et Hive n'ont pas toujours de mappage direct, quelques types Hive ont plusieurs types de données valides pour la sérialisation. Pour sérialiser les données en tant que type de données autre que par défaut, utilisez cette propriété. Pour plus d'informations sur le mappage des types, consultez la page Type mapping (Mappage des types) d'Amazon Ion sur GitHub.

Par défaut, les colonnes binaires Hive sont sérialisées en tant que blobs Amazon Ion, mais elles peuvent également être sérialisées au format clob Amazon Ion (grand objet de caractères). L'exemple suivant montre comment sérialiser la colonne `example_hive_binary_column` au format clob.

```
'ion.example_hive_binary_column.serialize_as' = 'clob'
```

Utilisation d'extracteurs de chemin

Amazon Ion est un format de fichier de style document, mais Apache Hive est un format à colonnes plates. Vous pouvez utiliser les SerDe propriétés spéciales d'Amazon Ion appelées `path`

`extractors` pour établir une correspondance entre les deux formats. Les extracteurs de chemins aplatissent le format hiérarchique Amazon Ion, mappent les valeurs Amazon Ion aux colonnes Hive et peuvent être utilisés pour renommer des champs.

Athena peut générer les extracteurs pour vous, mais vous pouvez également définir vos propres extracteurs si nécessaire.

Extracteurs de chemin générés

Par défaut, Athena recherche des valeurs Amazon Ion de premier niveau qui correspondent aux noms de colonnes Hive et crée des extracteurs de chemin à l'exécution en fonction de ces valeurs correspondantes. Si votre format de données Amazon Ion correspond au schéma de la table Hive, Athena génère dynamiquement les extracteurs pour vous, et vous n'avez pas besoin d'ajouter d'autres extracteurs de chemin. Ces extracteurs de chemins par défaut ne sont pas stockés dans les métadonnées de la table.

L'exemple suivant montre comment Athena génère des extracteurs basés sur le nom de la colonne.

```
-- Example Amazon Ion Document
{
    identification: {
        name: "John Smith",
        driver_license: "XXXX"
    },

    alias: "Johnny"
}

-- Example DDL
CREATE EXTERNAL TABLE example_schema2 (
    identification MAP<STRING, STRING>,
    alias STRING
)
STORED AS ION
LOCATION 's3://DOC-EXAMPLE-BUCKET/path_extraction1/'
```

Les exemples d’extracteurs suivants sont générés par Athena. Le premier extrait le champ `identification` vers la colonne `identification` et le second extrait le champ `alias` vers la colonne `alias`.

```
'ion.identification.path_extractor' = '(identification)'
```

```
'ion.alias.path_extractor' = '(alias)'
```

L'exemple suivant montre la table extraite.

```
|                  identification                    |  alias   |
|----------------------------------------------------|----------|
|{["name", "driver_license"],["John Smith", "XXXX"]} | "Johnny" |
```

Spécification de vos propres extracteurs de chemin

Si vos champs Amazon Ion ne sont pas parfaitement mappés avec les colonnes Hive, vous pouvez spécifier vos propres extracteurs de chemin. Dans la clause `WITH SERDEPROPERTIES` de votre instruction `CREATE TABLE`, utilisez la syntaxe suivante.

```
WITH SERDEPROPERTIES (
   "ion.path_extractor.case_sensitive" = "<Boolean>",
   "ion.<column_name>.path_extractor" = "<path_extractor_expression>"
)
```

> **Note**
>
> Par défaut, les extracteurs de chemins ne sont pas sensibles à la casse. Pour annuler ce paramètre, définissez la ion.path_extractor.case_sensitive SerDe propriété sur. `true`

Utilisation des chemins de recherche dans les extracteurs de chemins

<path_extractor_expression>La syntaxe des SerDe propriétés de l'extracteur de chemin contient :

```
"ion.<column_name>.path_extractor" = "<path_extractor_expression>"
```

Vous pouvez utiliser la *<path_extractor_expression>* pour spécifier un chemin de recherche qui analyse le document Amazon Ion et trouve les données correspondantes. Le chemin de recherche est entre parenthèses et peut contenir un ou plusieurs des composants suivants séparés par des espaces.

- Wild card – Correspond à toutes les valeurs.
- Index – Correspond à la valeur de l'index numérique spécifié. Les index sont basés sur zéro.

- Text – Correspond à toutes les valeurs dont les noms de champs correspondants sont équivalents au texte spécifié.
- Annotations – Correspond aux valeurs spécifiées par un composant de chemin encapsulé dont les annotations sont spécifiées.

L'exemple suivant montre un document Amazon Ion et quelques exemples de chemins de recherche.

```
-- Amazon Ion document
{
    foo: ["foo1", "foo2"] ,
    bar: "myBarValue",
    bar: A::"annotatedValue"
}

-- Example search paths
(foo 0)       # matches "foo1"
(1)           # matches "myBarValue"
(*)           # matches ["foo1", "foo2"], "myBarValue" and A::"annotatedValue"
()            # matches {foo: ["foo1", "foo2"] , bar: "myBarValue", bar:
 A::"annotatedValue"}
(bar)         # matches "myBarValue" and A::"annotatedValue"
(A::bar)      # matches A::"annotatedValue"
```

Exemples d'extracteurs

Aplatissement et renommage des champs

L'exemple suivant montre un ensemble de chemins de recherche qui aplatissent et renomment les champs. L'exemple utilise des chemins de recherche pour effectuer les opérations suivantes :

- Mapper la colonne `nickname` au champ `alias`
- Mapper la colonne `name` au sous-champ `name` situé dans le struct `identification`.

Voici l'exemple de document Amazon Ion.

```
-- Example Amazon Ion Document
{
    identification: {
        name: "John Smith",
        driver_license: "XXXX"
```

```
    },

    alias: "Johnny"
}
```

Voici l'exemple suivant d'instruction `CREATE TABLE` qui définit les extracteurs de chemin.

```
-- Example DDL Query
CREATE EXTERNAL TABLE example_schema2 (
    name STRING,
    nickname STRING
)
ROW FORMAT SERDE
 'com.amazon.ionhiveserde.IonHiveSerDe'
WITH SERDEPROPERTIES (
 'ion.nickname.path_extractor' = '(alias)',
 'ion.name.path_extractor' = '(identification name)'
 )
STORED AS ION
LOCATION 's3://DOC-EXAMPLE-BUCKET/path_extraction2/'
```

L'exemple suivant montre les données extraites.

```
-- Extracted Table
| name         |   nickname   |
|--------------|--------------|
| "John Smith" |  "Johnny"    |
```

Pour plus d'informations sur les chemins de recherche et d'autres exemples de chemins de recherche, consultez la page Ion Java Path Extraction sur GitHub.

Extraction des données de vol au format texte

L'exemple de requête `CREATE TABLE` suivant utilise `WITH SERDEPROPERTIES` pour ajouter des extracteurs de chemin pour extraire les données de vol et spécifier le codage de sortie sous forme de texte Amazon Ion. L'exemple utilise la syntaxe `STORED AS ION`.

```
CREATE EXTERNAL TABLE flights_ion (
    yr INT,
    quarter INT,
    month INT,
    dayofmonth INT,
```

```
    dayofweek INT,
    flightdate STRING,
    uniquecarrier STRING,
    airlineid INT,
)
ROW FORMAT SERDE
 'com.amazon.ionhiveserde.IonHiveSerDe'
WITH SERDEPROPERTIES (
 'ion.encoding' = 'TEXT',
 'ion.yr.path_extractor'='(year)',
 'ion.quarter.path_extractor'='(results quarter)',
 'ion.month.path_extractor'='(date month)')
STORED AS ION
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
```

SerDe Avro

Nom du SerDe

SerDe Avro

Nom de bibliothèque

org.apache.hadoop.hive.serde2.avro.AvroSerDe

Exemples

Pour des raisons de sécurité, Athena ne prend pas en charge l'utilisation de `avro.schema.url` dans la spécification du schéma d'une table. Utilisez `avro.schema.literal`. Pour extraire le schéma à partir d'un fichier Avro, vous pouvez utiliser le fichier Apache `avro-tools-<version>.jar` avec le paramètre `getschema`. Un schéma que vous pouvez utiliser dans votre instruction `WITH SERDEPROPERTIES` est alors renvoyé. Par exemple :

```
java -jar avro-tools-1.8.2.jar getschema my_data.avro
```

Le fichier `avro-tools-<version>.jar` se trouve dans le sous-répertoire `java` de votre version Avro installée. Pour télécharger Avro, consultez les versions d'Apache Avro. Pour télécharger Apache Avro Tools directement, consultez Apache Avro Tools Maven Repository.

Après avoir obtenu le schéma, utilisez une instruction `CREATE TABLE` pour créer une table Athena basée sur les données Avro sous-jacentes stockées dans Simple Storage Service (Amazon S3). Pour spécifier le SerDe Avro, utilisez `ROW FORMAT SERDE`

`'org.apache.hadoop.hive.serde2.avro.AvroSerDe'`. Comme indiqué dans l'exemple suivant, vous devez spécifier le schéma à l'aide de la clause `WITH SERDEPROPERTIES` en plus de spécifier les noms de colonne et les types de données correspondants pour la table.

> **Note**
>
> Remplacez *myregion* dans `s3://athena-examples-`*myregion*`/path/to/data/` par l'identifiant de région dans lequel vous exécutez Athena, par exemple, `s3://athena-examples-us-west-1/path/to/data/`.

```
CREATE EXTERNAL TABLE flights_avro_example (
   yr INT,
   flightdate STRING,
   uniquecarrier STRING,
   airlineid INT,
   carrier STRING,
   flightnum STRING,
   origin STRING,
   dest STRING,
   depdelay INT,
   carrierdelay INT,
   weatherdelay INT
)
PARTITIONED BY (year STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
WITH SERDEPROPERTIES ('avro.schema.literal'='
{
   "type" : "record",
   "name" : "flights_avro_subset",
   "namespace" : "default",
   "fields" : [ {
      "name" : "yr",
      "type" : [ "null", "int" ],
      "default" : null
   }, {
      "name" : "flightdate",
      "type" : [ "null", "string" ],
      "default" : null
   }, {
      "name" : "uniquecarrier",
      "type" : [ "null", "string" ],
```

```
      "default" : null
   }, {
      "name" : "airlineid",
      "type" : [ "null", "int" ],
      "default" : null
   }, {
      "name" : "carrier",
      "type" : [ "null", "string" ],
      "default" : null
   }, {
      "name" : "flightnum",
      "type" : [ "null", "string" ],
      "default" : null
   }, {
      "name" : "origin",
      "type" : [ "null", "string" ],
      "default" : null
   }, {
      "name" : "dest",
      "type" : [ "null", "string" ],
      "default" : null
   }, {
      "name" : "depdelay",
      "type" : [ "null", "int" ],
      "default" : null
   }, {
      "name" : "carrierdelay",
      "type" : [ "null", "int" ],
      "default" : null
   }, {
      "name" : "weatherdelay",
      "type" : [ "null", "int" ],
      "default" : null
    } ]
}
')
STORED AS AVRO
LOCATION 's3://athena-examples-myregion/flight/avro/';
```

Exécutez l'instruction `MSCK REPAIR TABLE` sur la table pour actualiser les métadonnées de la partition.

```
MSCK REPAIR TABLE flights_avro_example;
```

Recherchez les 10 principales villes de départ par nombre total de départs.

```
SELECT origin, count(*) AS total_departures
FROM flights_avro_example
WHERE year >= '2000'
GROUP BY origin
ORDER BY total_departures DESC
LIMIT 10;
```

> **Note**
>
> Les données du tableau des vols proviennent des Vols fournis par le ministère américain des transports, Bureau of Transportation Statistics (Bureau des statistiques des transports). Désaturé de l'original.

Grok SerDe

Le Logstash Grok SerDe est une bibliothèque contenant un ensemble de modèles spécialisés pour la désérialisation de données textuelles non structurées, généralement des journaux. Chaque modèle Grok est une expression régulière nommée. Vous pouvez identifier et réutiliser ces modèles de désérialisation selon vos besoins. Il est ainsi plus facile d'utiliser Grok que des expressions régulières. Grok fournit un ensemble de modèles prédéfinis. Vous pouvez aussi créer des modèles personnalisés.

Pour spécifier le Grok SerDe lors de la création d'une table dans Athena, utilisez `ROW FORMAT SERDE 'com.amazonaws.glue.serde.GrokSerDe'` la clause, suivie de `WITH SERDEPROPERTIES` la clause qui spécifie les modèles à associer à vos données, où :

- L'expression `input.format` définit les modèles de correspondance du fichier de données. Elle est obligatoire.
- L'expression `input.grokCustomPatterns` définit un modèle personnalisé nommé, que vous pouvez ensuite utiliser au sein de l'expression `input.format`. Elle est facultative. Pour inclure plusieurs entrées dans le modèle d' expression `input.grokCustomPatterns`, utilisez le caractère d'échappement de saut de ligne (\n) pour les séparer, comme suit : `'input.grokCustomPatterns'='INSIDE_QS ([^\"]*)\nINSIDE_BRACKETS ([^\]]*)')`.
- Les clauses `STORED AS INPUTFORMAT` et `OUTPUTFORMAT` sont obligatoires.

- La clause `LOCATION` spécifie un compartiment Simple Storage Service (Amazon S3), qui peut contenir plusieurs objets de données. Tous les objets de données du compartiment sont désérialisés pour créer la table.

Exemples

Ces exemples s'appuient sur la liste des modèles Grok prédéfinis. Consultez Pre-defined patterns.

Exemple 1

Cet exemple utilise la source des données d'entrées maillog Postfix enregistrées dans `s3://DOC-EXAMPLE-BUCKET/groksample/`.

```
Feb  9 07:15:00 m4eastmail postfix/smtpd[19305]: B88C4120838: connect from
 unknown[192.168.55.4]
Feb  9 07:15:00 m4eastmail postfix/smtpd[20444]: B58C4330038:
 client=unknown[192.168.55.4]
Feb  9 07:15:03 m4eastmail postfix/cleanup[22835]: BDC22A77854: message-
id=<31221401257553.5004389LCBF@m4eastmail.example.com>
```

L'instruction suivante crée une table dans Athena appelée `mygroktable` depuis le fichier de données source, à l'aide d'un modèle personnalisé et des modèles prédéfinis que vous spécifiez :

```
CREATE EXTERNAL TABLE `mygroktable`(
   syslogbase string,
   queue_id string,
   syslog_message string
   )
ROW FORMAT SERDE
   'com.amazonaws.glue.serde.GrokSerDe'
WITH SERDEPROPERTIES (
   'input.grokCustomPatterns' = 'POSTFIX_QUEUEID [0-9A-F]{7,12}',
   'input.format'='%{SYSLOGBASE} %{POSTFIX_QUEUEID:queue_id}:
 %{GREEDYDATA:syslog_message}'
   )
STORED AS INPUTFORMAT
   'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
   'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
   's3://DOC-EXAMPLE-BUCKET/groksample/';
```

Commencez par un modèle simple, comme `%{NOTSPACE:column}`, pour obtenir les colonnes mappées en premier, puis procédez à la spécialisation des colonnes si vous le souhaitez.

Exemple 2

Dans l'exemple suivant, vous créez une requête pour les journaux Log4j. Le format des entrées de l'exemple de journal est le suivant :

```
2017-09-12 12:10:34,972 INFO  - processType=AZ, processId=ABCDEFG614B6F5E49,
 status=RUN,
threadId=123:amqListenerContainerPool23P:AJ|ABCDE9614B6F5E49||
2017-09-12T12:10:11.172-0700],
executionTime=7290, tenantId=12456, userId=123123f8535f8d76015374e7a1d87c3c,
 shard=testapp1,
jobId=12312345e5e7df0015e777fb2e03f3c, messageType=REAL_TIME_SYNC,
action=receive, hostname=1.abc.def.com
```

Pour interroger les données des journaux :

- Ajoutez le modèle Grok au format `input.format` pour chaque colonne. Par exemple, pour `timestamp`, ajoutez `%{TIMESTAMP_ISO8601:timestamp}`. Pour `loglevel`, ajoutez `%{LOGLEVEL:loglevel}`.
- Assurez-vous que le modèle défini dans `input.format` correspond exactement au format du journal, en mappant les tirets (`-`) et les virgules qui séparent les entrées dans le format du journal.

```
CREATE EXTERNAL TABLE bltest (
 timestamp STRING,
 loglevel STRING,
 processtype STRING,
 processid STRING,
 status STRING,
 threadid STRING,
 executiontime INT,
 tenantid INT,
 userid STRING,
 shard STRING,
 jobid STRING,
 messagetype STRING,
 action STRING,
 hostname STRING
 )
```

```
ROW FORMAT SERDE 'com.amazonaws.glue.serde.GrokSerDe'
WITH SERDEPROPERTIES (
"input.grokCustomPatterns" = 'C_ACTION receive|send',
"input.format" = "%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:loglevel} - processType=
%{NOTSPACE:processtype}, processId=%{NOTSPACE:processid}, status=%{NOTSPACE:status},
 threadId=%{NOTSPACE:threadid}, executionTime=%{POSINT:executiontime}, tenantId=
%{POSINT:tenantid}, userId=%{NOTSPACE:userid}, shard=%{NOTSPACE:shard}, jobId=
%{NOTSPACE:jobid}, messageType=%{NOTSPACE:messagetype}, action=%{C_ACTION:action},
 hostname=%{HOST:hostname}"
) STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/samples/';
```

Exemple 3

L'exemple suivant d'interrogation des journaux Simple Storage Service (Amazon S3) montre l'expression `'input.grokCustomPatterns'` qui contient deux entrées de modèle, séparées par le caractère d'échappement de saut de ligne (\n), comme illustré dans cet extrait de l'exemple de requête : `'input.grokCustomPatterns'='INSIDE_QS ([^\"]*)\nINSIDE_BRACKETS ([^ \\]]*)'`).

```
CREATE EXTERNAL TABLE `s3_access_auto_raw_02`(
  `bucket_owner` string COMMENT 'from deserializer',
  `bucket` string COMMENT 'from deserializer',
  `time` string COMMENT 'from deserializer',
  `remote_ip` string COMMENT 'from deserializer',
  `requester` string COMMENT 'from deserializer',
  `request_id` string COMMENT 'from deserializer',
  `operation` string COMMENT 'from deserializer',
  `key` string COMMENT 'from deserializer',
  `request_uri` string COMMENT 'from deserializer',
  `http_status` string COMMENT 'from deserializer',
  `error_code` string COMMENT 'from deserializer',
  `bytes_sent` string COMMENT 'from deserializer',
  `object_size` string COMMENT 'from deserializer',
  `total_time` string COMMENT 'from deserializer',
  `turnaround_time` string COMMENT 'from deserializer',
  `referrer` string COMMENT 'from deserializer',
  `user_agent` string COMMENT 'from deserializer',
  `version_id` string COMMENT 'from deserializer')
ROW FORMAT SERDE
  'com.amazonaws.glue.serde.GrokSerDe'
```

```
WITH SERDEPROPERTIES (
  'input.format'='%{NOTSPACE:bucket_owner} %{NOTSPACE:bucket} \
\[%{INSIDE_BRACKETS:time}\\] %{NOTSPACE:remote_ip} %{NOTSPACE:requester}
 %{NOTSPACE:request_id} %{NOTSPACE:operation} %{NOTSPACE:key} \"?
%{INSIDE_QS:request_uri}\"? %{NOTSPACE:http_status} %{NOTSPACE:error_code}
 %{NOTSPACE:bytes_sent} %{NOTSPACE:object_size} %{NOTSPACE:total_time}
 %{NOTSPACE:turnaround_time} \"?%{INSIDE_QS:referrer}\"? \"?%{INSIDE_QS:user_agent}\"?
 %{NOTSPACE:version_id}',
  'input.grokCustomPatterns'='INSIDE_QS ([^\"]*)\nINSIDE_BRACKETS ([^\\]]*)')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET'
```

SerDe bibliothèques JSON

Dans Athena, vous pouvez utiliser des SerDe bibliothèques pour désérialiser les données JSON. La désérialisation convertit les données JSON afin qu'elles puissent être sérialisées (écrites) dans un format différent comme Parquet ou ORC.

- La bibliothèque Hive JSON SerDe native
- L'interface OpenX JSON SerDe
- L'interface SerDe Amazon Ion Hive

> **Note**
>
> Les bibliothèques Hive et OpenX s'attendent à ce que les données JSON soient sur une seule ligne (non formatées), les registres étant séparés par un caractère de nouvelle ligne. L'Amazon Ion Hive SerDe ne répond pas à cette exigence et peut être utilisé comme alternative car le format de données Ion est un sur-ensemble de JSON.

Noms des bibliothèques

Utilisez l'une des options suivantes :

org.apache.hive.hcatalog.data. JsonSerDe

org.openx.data.json. JsonSerDe

com.amazon.ionhiveserde. IonHiveSerDe

Hive JSON SerDe

Le Hive JSON SerDe est couramment utilisé pour traiter des données JSON comme des événements. Ces événements sont représentés sous forme de chaînes sur une ligne de texte codées en JSON et séparées par une nouvelle ligne. Le JSON Hive n' SerDe autorise pas la duplication de clés `map` ou de noms de `struct` clés.

> **Note**
>
> Il SerDe s'attend à ce que chaque document JSON se trouve sur une seule ligne de texte sans aucun caractère de fin de ligne séparant les champs de l'enregistrement. Si le texte JSON est dans un joli format d'impression, vous pouvez recevoir un message d'erreur tel que HIVE_CURSOR_ERROR : Row is not a valid JSON Object ou HIVE_CURSOR_ERROR : : Unexpected JsonParseException end-of-input : expected close marker for OBJECT lorsque vous essayez d'interroger la table après l'avoir créée. Pour plus d'informations, consultez la section Fichiers de données JSON dans la SerDe documentation OpenX sur. GitHub

L'exemple d'instruction DDL suivant utilise le code JSON Hive SerDe pour créer un tableau basé sur des exemples de données publicitaires en ligne. Dans la clause `LOCATION`, remplacez *`myregion`* dans `s3://DOC-EXAMPLE-BUCKET.elasticmapreduce/samples/hive-ads/tables/impressions` par l'identifiant de la région où vous exécutez Athena (par exemple, `s3://us-west-2.elasticmapreduce/samples/hive-ads/tables/impressions`).

```
CREATE EXTERNAL TABLE impressions (
    requestbegintime string,
    adid string,
    impressionid string,
    referrer string,
    useragent string,
    usercookie string,
    ip string,
    number string,
    processid string,
    browsercookie string,
    requestendtime string,
```

```
    timers struct
                <
                 modellookup:string,
                 requesttime:string
                >,
    threadid string,
    hostname string,
    sessionid string
)
PARTITIONED BY (dt string)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET.elasticmapreduce/samples/hive-ads/tables/
impressions';
```

Spécifier les formats d'horodatage avec le Hive JSON SerDe

Pour analyser les valeurs de l'horodatage à partir d'une chaîne, vous pouvez ajouter le sous-champ `WITH SERDEPROPERTIES` à la clause `ROW FORMAT SERDE` et l'utiliser pour spécifier le paramètre `timestamp.formats`. Dans le paramètre, spécifiez une liste séparée par des virgules d'un ou plusieurs modèles d'horodatage, comme dans l'exemple suivant :

```
...
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
WITH SERDEPROPERTIES ("timestamp.formats"="yyyy-MM-dd'T'HH:mm:ss.SSS'Z',yyyy-MM-
dd'T'HH:mm:ss")
...
```

Pour en savoir plus, consultez la rubrique Horodatage dans la documentation Apache Hive.

Chargement de la table pour l'interrogation

Après avoir créé la table, exécutez MSCK REPAIR TABLE pour charger la table et la rendre interrogeable à partir d'Athena :

```
MSCK REPAIR TABLE impressions
```

Journaux d'interrogation CloudTrail

Vous pouvez utiliser le JSON Hive SerDe pour interroger les CloudTrail journaux. Pour plus d'informations et des exemples d'instructions `CREATE TABLE`, voir Journaux d'interrogation AWS CloudTrail.

OpenX JSON SerDe

Comme le JSON Hive SerDe, vous pouvez utiliser le JSON OpenX pour traiter les données JSON. Les données sont également représentées sous forme de chaînes sur une ligne de texte codées en JSON et séparées par une nouvelle ligne. Comme le JSON Hive SerDe, le JSON OpenX SerDe n'autorise pas la duplication de clés `struct` ou de noms `map` de clés.

> **Note**
>
> Il SerDe s'attend à ce que chaque document JSON se trouve sur une seule ligne de texte sans aucun caractère de fin de ligne séparant les champs de l'enregistrement. Si le texte JSON est dans un joli format d'impression, vous pouvez recevoir un message d'erreur tel que HIVE_CURSOR_ERROR : Row is not a valid JSON Object ou HIVE_CURSOR_ERROR : : Unexpected JsonParseException end-of-input : expected close marker for OBJECT lorsque vous essayez d'interroger la table après l'avoir créée. Pour plus d'informations, consultez la section Fichiers de données JSON dans la SerDe documentation OpenX sur. GitHub

Propriétés facultatives

Contrairement au JSON Hive SerDe, le JSON OpenX SerDe possède également les propriétés SerDe facultatives suivantes qui peuvent être utiles pour corriger les incohérences dans les données.

ignore.malformed.json

Facultatif. Lorsque la valeur est définie sur `TRUE`, cela vous permet d'ignorer la syntaxe JSON incorrecte. L'argument par défaut est `FALSE`.

dots.in.keys

Facultatif. L'argument par défaut est `FALSE`. Lorsqu'il est défini sur`TRUE`, permet de remplacer SerDe les points dans les noms des clés par des traits de soulignement. Par exemple, si le jeu de données JSON contient une clé portant le nom `"a.b"`, vous pouvez utiliser cette propriété pour définir le nom de la colonne comme étant `"a_b"` dans Athena. Par défaut (sans cela SerDe), Athena n'autorise pas les points dans les noms de colonnes.

case.insensitive

Facultatif. L'argument par défaut est `TRUE`. Lorsqu'il est défini sur`TRUE`, il SerDe convertit toutes les colonnes majuscules en minuscules.

Pour utiliser des noms de clés sensibles à la casse dans vos données, utilisez `WITH SERDEPROPERTIES ("case.insensitive"= FALSE;)`. Ensuite, pour chaque clé qui n'est pas encore entièrement en minuscules, fournissez un mappage entre le nom de la colonne et le nom de la propriété à l'aide de la syntaxe suivante :

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("case.insensitive" = "FALSE", "mapping.userid" = "userId")
```

Si vous avez deux clés, comme `URL` et `Url`, qui sont identiques quand elles sont écrites en minuscules, une erreur comme celle-ci peut se produire :

HIVE_CURSOR_ERROR: Row is not a valid JSON Object - JSONException: Duplicate key "url"

Pour résoudre ce problème, définissez la propriété `case.insensitive` sur `FALSE` et mapper les clés avec différents noms, comme dans l'exemple suivant :

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("case.insensitive" = "FALSE", "mapping.url1" = "URL",
 "mapping.url2" = "Url")
```

mappage

Facultatif. Cette propriété mappe les noms de colonnes aux clés JSON qui ne sont pas identiques aux noms de colonne. Le paramètre `mapping` est utile lorsque les données JSON contiennent des clés qui sont des mots-clés. Par exemple, si vous avez une clé JSON nommée `timestamp`, utilisez la syntaxe suivante pour mapper la clé à une colonne nommée `ts`:

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("mapping.ts" = "timestamp")
```

Mapper les noms de champs imbriqués avec des deux-points à des noms compatibles avec Hive

Si vous avez un nom de champ avec des deux-points à l'intérieur d'un `struct`, vous pouvez utiliser la propriété `mapping` pour associer le champ à un nom compatible avec Hive. Par exemple, si vos définitions de type de colonne contiennent `my:struct:field:string`, vous pouvez mapper la définition à `my_struct_field:string` en incluant l'entrée suivante dans `WITH SERDEPROPERTIES` :

```
("mapping.my_struct_field" = "my:struct:field")
```

L’exemple suivant montre l’instruction `CREATE TABLE` correspondante.

```
CREATE EXTERNAL TABLE colon_nested_field (
item struct<my_struct_field:string>)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("mapping.my_struct_field" = "my:struct:field")
```

Exemple : données publicitaires

L'exemple d'instruction DDL suivant utilise le code SerDe JSON OpenX pour créer un tableau basé sur les mêmes exemples de données publicitaires en ligne que ceux utilisés dans l'exemple du fichier JSON Hive. SerDe Dans la clause `LOCATION`, remplacez *`myregion`* par l'identifiant de la région dans laquelle vous exécutez Athena.

```
CREATE EXTERNAL TABLE impressions (
    requestbegintime string,
    adid string,
    impressionId string,
    referrer string,
    useragent string,
    usercookie string,
    ip string,
    number string,
    processid string,
    browsercokie string,
    requestendtime string,
    timers struct<
       modellookup:string,
       requesttime:string>,
    threadid string,
    hostname string,
    sessionid string
)   PARTITIONED BY (dt string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET.elasticmapreduce/samples/hive-ads/tables/
impressions';
```

Exemple : désérialisation des données JSON imbriquées

Vous pouvez utiliser le JSON SerDes pour analyser des données plus complexes codées en JSON. Cela implique l'utilisation d'instructions `CREATE TABLE` utilisant des éléments `struct` et `array` pour représenter des structures imbriquées.

L'exemple suivant crée une table Athena à partir de données JSON ayant des structures imbriquées. L'exemple présente la structure suivante :

```
{
"DocId": "AWS",
"User": {
        "Id": 1234,
        "Username": "carlos_salazar",
        "Name": "Carlos",
"ShippingAddress": {
"Address1": "123 Main St.",
"Address2": null,
"City": "Anytown",
"State": "CA"
   },
"Orders": [
   {
     "ItemId": 6789,
     "OrderDate": "11/11/2022"
   },
   {
     "ItemId": 4352,
     "OrderDate": "12/12/2022"
   }
  ]
 }
}
```

N'oubliez pas qu'OpenX SerDe s'attend à ce que chaque enregistrement JSON se trouve sur une seule ligne de texte. Lorsqu'elles sont stockées dans Amazon S3, toutes les données de l'exemple précédent doivent se trouver sur une seule ligne, comme ceci :

```
{"DocId":"AWS","User":
{"Id":1234,"Username":"carlos_salazar","Name":"Carlos","ShippingAddress" ...
```

L'`CREATE TABLE`instruction suivante utilise le Openx- JsonSerDe avec les types de données de `array` collecte `struct` et pour établir des groupes d'objets pour les données d'exemple.

```
CREATE external TABLE complex_json (
   docid string,
   `user` struct<
               id:INT,
               username:string,
               name:string,
               shippingaddress:struct<
                                address1:string,
                                address2:string,
                                city:string,
                                state:string
                                >,
               orders:array<
                           struct<
                                itemid:INT,
                                 orderdate:string
                                 >
                             >
               >
   )
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/myjsondata/';
```

Pour interroger la table, utilisez une `SELECT` instruction comme celle-ci.

```
SELECT
 user.name as Name,
 user.shippingaddress.address1 as Address,
 user.shippingaddress.city as City,
 o.itemid as Item_ID, o.orderdate as Order_date
FROM complex_json, UNNEST(user.orders) as temp_table (o)
```

Pour accéder aux champs de données à l'intérieur des structures, l'exemple de requête utilise la notation par points (par exemple,`user.name`). Pour accéder aux données d'un tableau de structures (comme pour le `orders` champ), vous pouvez utiliser la `UNNEST` fonction. La `UNNEST` fonction aplatit le tableau pour en faire une table temporaire (appelée o dans ce cas). Cela vous permet d'utiliser la notation par points comme vous le feriez avec les structures pour accéder aux éléments de tableau

non imbriqués (par exemple,). `o.itemid` Le nom`temp_table`, utilisé dans l'exemple à des fins d'illustration, est souvent abrégé en. `t`

Le tableau suivant présente les résultats de la requête.

#	Nom	Address	Ville	Identifiant de l'article	Date_commande
1	Carlos	123 rue Main	Anytown	6789	11/11/2022
2	Carlos	123 rue Main	Anytown	4352	12/12/2022

Ressources supplémentaires

Pour de plus amples informations sur l'utilisation de JSON et de JSON imbriqué dans Athena, consultez les ressources suivantes :

- Création de tables dans Amazon Athena à partir de JSON imbriqué et de mappages à l'aide de JSON SerDe (AWS Big Data Blog)
- Je reçois des erreurs lorsque j'essaie de lire des données JSON dans Amazon Athena (article du AWS Knowledge Center)
- hive-json-schema(GitHub) — Outil écrit en Java qui génère des `CREATE TABLE` instructions à partir d'exemples de documents JSON. Les instructions `CREATE TABLE` générées utilisent le SerDe JSON OpenX.

LazySimpleSerDe pour les fichiers CSV, TSV et délimités de manière personnalisée

Cette spécification SerDe est facultative. C'est le cas SerDe pour les données au format CSV, TSV et aux formats délimités personnalisés qu'Athena utilise par défaut. Ceci SerDe est utilisé si vous n'en spécifiez aucune SerDe et que vous spécifiez uniquement`ROW FORMAT DELIMITED`. Utilisez-le SerDe si vos données ne comportent pas de valeurs entre guillemets.

Pour une documentation de référence sur le LazySimpleSerDe, consultez la SerDe section Hive du guide du développeur Apache Hive.

Nom de bibliothèque

Le nom de la bibliothèque de classes pour le LazySimpleSerDe est`org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`. Pour plus d'informations sur la LazySimpleSerDe classe, consultez LazySimpleSerDe.java sur GitHub .com.

Omission des en-têtes

Pour ignorer les en-têtes dans vos données lorsque vous définissez une table, vous pouvez utiliser la propriété de table `skip.header.line.count`, comme dans l'exemple suivant.

```
TBLPROPERTIES ("skip.header.line.count"="1")
```

Pour des exemples, voir les instructions `CREATE TABLE` dans Interrogation des journaux de flux Amazon VPC et Interrogation des journaux Amazon CloudFront .

Exemple CSV

L'exemple suivant montre comment utiliser le `LazySimpleSerDe` pour créer une table dans Athena à partir de données CSV. Pour désérialiser des fichiers délimités de manière personnalisée à l'aide de cette méthode SerDe, suivez le modèle décrit dans les exemples, mais utilisez la `FIELDS TERMINATED BY` clause pour spécifier un autre délimiteur à caractère unique. LazySimpleSerDe ne prend pas en charge les délimiteurs à plusieurs caractères.

> **Note**
>
> Remplacez *`myregion`* dans `s3://athena-examples-`*`myregion`*`/path/to/data/` par l'identifiant de région dans lequel vous exécutez Athena, par exemple, `s3://athena-examples-us-west-1/path/to/data/`.

Utilisez l'instruction `CREATE TABLE` pour créer une table Athena à partir données CSV sous-jacentes stockées dans Simple Storage Service (Amazon S3).

```
CREATE EXTERNAL TABLE flight_delays_csv (
    yr INT,
    quarter INT,
    month INT,
    dayofmonth INT,
    dayofweek INT,
    flightdate STRING,
```

```
    uniquecarrier STRING,
    airlineid INT,
    carrier STRING,
    tailnum STRING,
    flightnum STRING,
    originairportid INT,
    originairportseqid INT,
    origincitymarketid INT,
    origin STRING,
    origincityname STRING,
    originstate STRING,
    originstatefips STRING,
    originstatename STRING,
    originwac INT,
    destairportid INT,
    destairportseqid INT,
    destcitymarketid INT,
    dest STRING,
    destcityname STRING,
    deststate STRING,
    deststatefips STRING,
    deststatename STRING,
    destwac INT,
    crsdeptime STRING,
    deptime STRING,
    depdelay INT,
    depdelayminutes INT,
    depdel15 INT,
    departuredelaygroups INT,
    deptimeblk STRING,
    taxiout INT,
    wheelsoff STRING,
    wheelson STRING,
    taxiin INT,
    crsarrtime INT,
    arrtime STRING,
    arrdelay INT,
    arrdelayminutes INT,
    arrdel15 INT,
    arrivaldelaygroups INT,
    arrtimeblk STRING,
    cancelled INT,
    cancellationcode STRING,
    diverted INT,
```

```
    crselapsedtime INT,
    actualelapsedtime INT,
    airtime INT,
    flights INT,
    distance INT,
    distancegroup INT,
    carrierdelay INT,
    weatherdelay INT,
    nasdelay INT,
    securitydelay INT,
    lateaircraftdelay INT,
    firstdeptime STRING,
    totaladdgtime INT,
    longestaddgtime INT,
    divairportlandings INT,
    divreacheddest INT,
    divactualelapsedtime INT,
    divarrdelay INT,
    divdistance INT,
    div1airport STRING,
    div1airportid INT,
    div1airportseqid INT,
    div1wheelson STRING,
    div1totalgtime INT,
    div1longestgtime INT,
    div1wheelsoff STRING,
    div1tailnum STRING,
    div2airport STRING,
    div2airportid INT,
    div2airportseqid INT,
    div2wheelson STRING,
    div2totalgtime INT,
    div2longestgtime INT,
    div2wheelsoff STRING,
    div2tailnum STRING,
    div3airport STRING,
    div3airportid INT,
    div3airportseqid INT,
    div3wheelson STRING,
    div3totalgtime INT,
    div3longestgtime INT,
    div3wheelsoff STRING,
    div3tailnum STRING,
    div4airport STRING,
```

```
    div4airportid INT,
    div4airportseqid INT,
    div4wheelson STRING,
    div4totalgtime INT,
    div4longestgtime INT,
    div4wheelsoff STRING,
    div4tailnum STRING,
    div5airport STRING,
    div5airportid INT,
    div5airportseqid INT,
    div5wheelson STRING,
    div5totalgtime INT,
    div5longestgtime INT,
    div5wheelsoff STRING,
    div5tailnum STRING
)
    PARTITIONED BY (year STRING)
    ROW FORMAT DELIMITED
      FIELDS TERMINATED BY ','
      ESCAPED BY '\\'
      LINES TERMINATED BY '\n'
    LOCATION 's3://athena-examples-myregion/flight/csv/';
```

Exécutez l'instruction `MSCK REPAIR TABLE` pour actualiser les métadonnées de la partition à chaque ajout d'une nouvelle partition à cette table :

```
MSCK REPAIR TABLE flight_delays_csv;
```

Recherchez les 10 principaux itinéraires dont le retard dépasse 1 heure :

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_csv
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

> **Note**
>
> Les données du tableau des vols proviennent des Vols fournis par le ministère américain des transports, Bureau of Transportation Statistics (Bureau des statistiques des transports). Désaturé de l'original.

Exemple TSV

Pour créer une table Athena à partir de données TSV stockées dans Amazon S3, utilisez `ROW FORMAT DELIMITED` et spécifiez le `\t` comme délimiteur de champs de tabulation, `\n` comme séparateur de ligne et `\` comme caractère d'échappement. L'extrait suivant montre cette syntaxe. Aucun exemple de données de vol TSV n'est disponible sur l'emplacement `athena-examples`, mais comme dans le cas de la table CSV, vous devez exécuter `MSCK REPAIR TABLE` pour actualiser les métadonnées de partition chaque fois qu'une nouvelle partition est ajoutée.

```
...
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
...
```

SerDe OpenCSV pour le traitement des fichiers CSV

Lorsque vous créez une table Athena pour les données CSV, déterminez celle SerDe à utiliser en fonction des types de valeurs que contiennent vos données :

- Si vos données contiennent des valeurs entre guillemets (''), vous pouvez utiliser l' SerDeOpenCSV pour désérialiser les valeurs dans Athena. Si vos données ne contiennent pas de valeurs entre guillemets (''), vous pouvez ne pas SerDe en spécifier. Dans ce cas, Athena utilise le `LazySimpleSerDe` par défaut. Pour plus d'informations, veuillez consulter LazySimpleSerDe pour les fichiers CSV, TSV et délimités de manière personnalisée.
- Si vos données contiennent des `TIMESTAMP` valeurs numériques UNIX (par exemple,`1579059880000`), utilisez l'OpenCSV. SerDe Si vos données utilisent ce `java.sql.Timestamp` format, utilisez le LazySimpleSerDe.

CSV SerDe (OpenCSVSerDe)

L' SerDeOpenCSV présente les caractéristiques suivantes pour les données sous forme de chaîne :

- Utilise des guillemets (') en tant que caractère par défaut et vous permet de spécifier des séparateurs, des guillemets et des caractères d'échappement, tels que :

```
WITH SERDEPROPERTIES ("separatorChar" = ",", "quoteChar" = "`", "escapeChar" = "\\" )
```

- `\t` ou `\n` ne peuvent pas faire directement l'objet d'un échappement. Pour ce faire, utilisez `"escapeChar" = "\\"`. Consultez l'exemple de cette rubrique.
- Il ne prend pas en charge les sauts de ligne imbriqués dans des fichiers CSV.

Pour les types de données autres que`STRING`, l'SerDe OpenCSV se comporte comme suit :

- Reconnaît les de données `BOOLEAN`, `BIGINT`, `INT` et `DOUBLE`.
- Ne reconnaît pas les valeurs vides ou nulles dans les colonnes définies en tant que type de données numériques, et les conserve en tant que `string`. Une solution consiste à créer la colonne avec les valeurs nulles en tant que `string` et utiliser ensuite `CAST` pour convertir le champ dans une requête en un type de données numérique, en fournissant une valeur par défaut de `0` pour les valeurs nulles. Pour plus d'informations, voir Lorsque j'interroge des données CSV dans Athena, l'erreur HIVE_BAD_DATA s'affiche : Erreur lors de l'analyse de la valeur du champ dans le Knowledge Center. AWS
- Pour les colonnes spécifiées avec le type de données `timestamp` dans votre instruction `CREATE TABLE`, il reconnaît les données `TIMESTAMP` si elles sont spécifiées dans le format numérique UNIX en millisecondes, par exemple, `1579059880000`.
  - L'SerDe OpenCSV ne prend pas en `TIMESTAMP` charge le format `java.sql.Timestamp` compatible JDBC, tel `"YYYY-MM-DD HH:MM:SS.fffffffff"` que (précision à 9 décimales).
- Pour les colonnes spécifiées avec le type de données `DATE` dans votre instruction `CREATE TABLE`, il reconnaît les valeurs comme des dates si les valeurs représentent le nombre de jours qui se sont écoulés depuis le 1er janvier 1970. Par exemple, la valeur `18276` dans une colonne avec le type de données `date` est rendue comme `2020-01-15` lorsqu'elle est interrogée. Dans ce format UNIX, chaque jour est considéré comme ayant 86 400 secondes.
  - L'SerDe OpenCSV ne prend directement en `DATE` charge aucun autre format. Pour traiter les données d'horodatage dans d'autres formats, vous pouvez définir la colonne comme `string` et ensuite utiliser les fonctions de conversion de temps pour retourner les résultats souhaités dans

votre requête SELECT. Pour plus d'informations, consultez l'article Lorsque j'interroge une table dans Amazon Athena, le résultat TIMESTAMP est vide dans le Centre de connaissances AWS.

- Pour mieux convertir les colonnes en le type souhaité dans une table, vous pouvez créer une vue sur la table et utiliser CAST pour la conversion.

Example Exemple : Utilisation des types TIMESTAMP et DATE spécifiés au format numérique UNIX.

Considérez les trois colonnes suivantes de données séparées par des virgules. Les valeurs de chaque colonne sont placées entre guillemets.

```
"unixvalue creationdate 18276 creationdatetime 1579059880000","18276","1579059880000"
```

L'instruction suivante crée une table dans Athena à partir de l'emplacement du compartiment Simple Storage Service (Amazon S3) spécifié.

```
CREATE EXTERNAL TABLE IF NOT EXISTS testtimestamp1(
 `profile_id` string,
 `creationdate` date,
 `creationdatetime` timestamp
 )
 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
 LOCATION 's3://DOC-EXAMPLE-BUCKET'
```

Ensuite, exécutez la requête suivante :

```
SELECT * FROM testtimestamp1
```

La requête renvoie le résultat suivant, affichant les données de date et d'heure :

```
profile_id                                                      creationdate
 creationdatetime
unixvalue creationdate 18276 creationdatetime 1579146280000     2020-01-15
 2020-01-15 03:44:40.000
```

Example Exemple : Échappement de **\t** ou **\n**

Examinez les données de test suivantes :

```
" \\t\\t\\n 123 \\t\\t\\n ",abc
```

```
" 456 ",xyz
```

L'instruction suivante crée une table dans Athena, en spécifiant que `"escapeChar" = "\\"`.

```
CREATE EXTERNAL TABLE test1 (
f1 string,
s2 string)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES ("separatorChar" = ",", "escapeChar" = "\\")
LOCATION 's3://DOC-EXAMPLE-BUCKET/dataset/test1/'
```

Ensuite, exécutez la requête suivante :

```
SELECT * FROM test1;
```

Elle renvoie ce résultat, en échappant correctement `\t` ou `\n` :

```
f1            s2
\t\t\n 123 \t\t\n            abc
456                          xyz
```

SerDe nom

CSV SerDe

Nom de bibliothèque

Pour l'utiliser SerDe, spécifiez son nom de classe complet après`ROW FORMAT SERDE`. Spécifiez également les délimiteurs à l'intérieur `SERDEPROPERTIES`, comme suit :

```
...
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = ",",
  "quoteChar"     = "`",
  "escapeChar"    = "\\"
)
```

Omission des en-têtes

Pour ignorer les en-têtes dans vos données lorsque vous définissez une table, vous pouvez utiliser la propriété de table `skip.header.line.count`, comme dans l'exemple suivant.

```
TBLPROPERTIES ("skip.header.line.count"="1")
```

Pour des exemples, voir les instructions `CREATE TABLE` dans Interrogation des journaux de flux Amazon VPC et Interrogation des journaux Amazon CloudFront .

Exemple

Cet exemple suppose que des données en format CSV sont enregistrées dans `s3://DOC-EXAMPLE-BUCKET/mycsv/` avec le contenu suivant :

```
"a1","a2","a3","a4"
"1","2","abc","def"
"a","a1","abc3","ab4"
```

Utilisez une instruction `CREATE TABLE` pour créer une table Athena basée sur les données. Faites référence à la classe SerDe OpenCSV `ROW FORMAT SERDE` après et spécifiez le séparateur de caractères, le guillemet et le caractère d'échappement, comme `WITH SERDEPROPERTIES` dans l'exemple suivant.

```
CREATE EXTERNAL TABLE myopencsvtable (
   col1 string,
   col2 string,
   col3 string,
   col4 string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
   'separatorChar' = ',',
   'quoteChar' = '"',
   'escapeChar' = '\\'
   )
STORED AS TEXTFILE
LOCATION 's3://DOC-EXAMPLE-BUCKET/mycsv/';
```

Interrogez toutes les valeurs de la table :

```
SELECT * FROM myopencsvtable;
```

La requête renvoie les valeurs suivantes :

```
col1     col2    col3    col4
```

```
----------------------------
a1      a2      a3      a4
1       2       abc     def
a       a1      abc3    ab4
```

ORC SerDe

SerDe nom

OrcSerDe

Nom de bibliothèque

Cette bibliothèque utilise la classe Apache Hive OrcSerde.java pour les données au format ORC. Celle-ci transmet l'objet depuis ORC vers le lecteur et depuis ORC vers l'enregistreur.

Exemples

> **Note**
>
> Remplacez *myregion* dans `s3://athena-examples-`*myregion*`/path/to/data/` par l'identifiant de région dans lequel vous exécutez Athena, par exemple, `s3://athena-examples-us-west-1/path/to/data/`.

L'exemple suivant crée une table pour les données de retard de vol dans ORC. La table comprend des partitions :

```
DROP TABLE flight_delays_orc;
CREATE EXTERNAL TABLE flight_delays_orc (
    yr INT,
    quarter INT,
    month INT,
    dayofmonth INT,
    dayofweek INT,
    flightdate STRING,
    uniquecarrier STRING,
    airlineid INT,
    carrier STRING,
    tailnum STRING,
    flightnum STRING,
    originairportid INT,
```

```
    originairportseqid INT,
    origincitymarketid INT,
    origin STRING,
    origincityname STRING,
    originstate STRING,
    originstatefips STRING,
    originstatename STRING,
    originwac INT,
    destairportid INT,
    destairportseqid INT,
    destcitymarketid INT,
    dest STRING,
    destcityname STRING,
    deststate STRING,
    deststatefips STRING,
    deststatename STRING,
    destwac INT,
    crsdeptime STRING,
    deptime STRING,
    depdelay INT,
    depdelayminutes INT,
    depdel15 INT,
    departuredelaygroups INT,
    deptimeblk STRING,
    taxiout INT,
    wheelsoff STRING,
    wheelson STRING,
    taxiin INT,
    crsarrtime INT,
    arrtime STRING,
    arrdelay INT,
    arrdelayminutes INT,
    arrdel15 INT,
    arrivaldelaygroups INT,
    arrtimeblk STRING,
    cancelled INT,
    cancellationcode STRING,
    diverted INT,
    crselapsedtime INT,
    actualelapsedtime INT,
    airtime INT,
    flights INT,
    distance INT,
    distancegroup INT,
```

```
    carrierdelay INT,
    weatherdelay INT,
    nasdelay INT,
    securitydelay INT,
    lateaircraftdelay INT,
    firstdeptime STRING,
    totaladdgtime INT,
    longestaddgtime INT,
    divairportlandings INT,
    divreacheddest INT,
    divactualelapsedtime INT,
    divarrdelay INT,
    divdistance INT,
    div1airport STRING,
    div1airportid INT,
    div1airportseqid INT,
    div1wheelson STRING,
    div1totalgtime INT,
    div1longestgtime INT,
    div1wheelsoff STRING,
    div1tailnum STRING,
    div2airport STRING,
    div2airportid INT,
    div2airportseqid INT,
    div2wheelson STRING,
    div2totalgtime INT,
    div2longestgtime INT,
    div2wheelsoff STRING,
    div2tailnum STRING,
    div3airport STRING,
    div3airportid INT,
    div3airportseqid INT,
    div3wheelson STRING,
    div3totalgtime INT,
    div3longestgtime INT,
    div3wheelsoff STRING,
    div3tailnum STRING,
    div4airport STRING,
    div4airportid INT,
    div4airportseqid INT,
    div4wheelson STRING,
    div4totalgtime INT,
    div4longestgtime INT,
    div4wheelsoff STRING,
```

```
    div4tailnum STRING,
    div5airport STRING,
    div5airportid INT,
    div5airportseqid INT,
    div5wheelson STRING,
    div5totalgtime INT,
    div5longestgtime INT,
    div5wheelsoff STRING,
    div5tailnum STRING
)
PARTITIONED BY (year String)
STORED AS ORC
LOCATION 's3://athena-examples-myregion/flight/orc/'
tblproperties ("orc.compress"="ZLIB");
```

Exécutez l'instruction `MSCK REPAIR TABLE` sur la table pour actualiser les métadonnées de la partition:

```
MSCK REPAIR TABLE flight_delays_orc;
```

Utilisez cette requête pour obtenir les 10 principaux itinéraires dont le retard dépasse 1 heure :

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_orc
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

Parquet SerDe

SerDe nom

ParquetHiveSerDe est utilisé pour les données stockées au format Parquet.

> **Note**
>
> Pour convertir des données au format Parquet, vous pouvez utiliser les requêtes CREATE TABLE AS SELECT (CTAS) . Pour plus d'informations, consultez Création d'une table à partir des résultats des requêtes (CTAS), Exemples de requêtes CTAS et Utilisation de CTAS et INSERT INTO pour ETL et l'analyse des données.

Nom de bibliothèque

Athena utilise la classe suivante lorsqu'il est nécessaire de désérialiser les données stockées au stockage Parquet : `org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe`

Exemple : interrogation d'un fichier stocké au format Parquet

> **Note**
>
> Remplacez *myregion* dans `s3://athena-examples-`*myregion*`/path/to/data/` par l'identifiant de région dans lequel vous exécutez Athena, par exemple, `s3://athena-examples-us-west-1/path/to/data/`.

Utilisez l'`CREATE TABLE`instruction suivante pour créer une table Athena à partir des données sous-jacentes stockées au format Parquet dans Amazon S3 :

```
CREATE EXTERNAL TABLE flight_delays_pq (
    yr INT,
    quarter INT,
    month INT,
    dayofmonth INT,
    dayofweek INT,
    flightdate STRING,
    uniquecarrier STRING,
    airlineid INT,
    carrier STRING,
    tailnum STRING,
    flightnum STRING,
    originairportid INT,
    originairportseqid INT,
    origincitymarketid INT,
    origin STRING,
    origincityname STRING,
    originstate STRING,
    originstatefips STRING,
    originstatename STRING,
    originwac INT,
    destairportid INT,
    destairportseqid INT,
    destcitymarketid INT,
    dest STRING,
```

```
    destcityname STRING,
    deststate STRING,
    deststatefips STRING,
    deststatename STRING,
    destwac INT,
    crsdeptime STRING,
    deptime STRING,
    depdelay INT,
    depdelayminutes INT,
    depdel15 INT,
    departuredelaygroups INT,
    deptimeblk STRING,
    taxiout INT,
    wheelsoff STRING,
    wheelson STRING,
    taxiin INT,
    crsarrtime INT,
    arrtime STRING,
    arrdelay INT,
    arrdelayminutes INT,
    arrdel15 INT,
    arrivaldelaygroups INT,
    arrtimeblk STRING,
    cancelled INT,
    cancellationcode STRING,
    diverted INT,
    crselapsedtime INT,
    actualelapsedtime INT,
    airtime INT,
    flights INT,
    distance INT,
    distancegroup INT,
    carrierdelay INT,
    weatherdelay INT,
    nasdelay INT,
    securitydelay INT,
    lateaircraftdelay INT,
    firstdeptime STRING,
    totaladdgtime INT,
    longestaddgtime INT,
    divairportlandings INT,
    divreacheddest INT,
    divactualelapsedtime INT,
    divarrdelay INT,
```

```
    divdistance INT,
    div1airport STRING,
    div1airportid INT,
    div1airportseqid INT,
    div1wheelson STRING,
    div1totalgtime INT,
    div1longestgtime INT,
    div1wheelsoff STRING,
    div1tailnum STRING,
    div2airport STRING,
    div2airportid INT,
    div2airportseqid INT,
    div2wheelson STRING,
    div2totalgtime INT,
    div2longestgtime INT,
    div2wheelsoff STRING,
    div2tailnum STRING,
    div3airport STRING,
    div3airportid INT,
    div3airportseqid INT,
    div3wheelson STRING,
    div3totalgtime INT,
    div3longestgtime INT,
    div3wheelsoff STRING,
    div3tailnum STRING,
    div4airport STRING,
    div4airportid INT,
    div4airportseqid INT,
    div4wheelson STRING,
    div4totalgtime INT,
    div4longestgtime INT,
    div4wheelsoff STRING,
    div4tailnum STRING,
    div5airport STRING,
    div5airportid INT,
    div5airportseqid INT,
    div5wheelson STRING,
    div5totalgtime INT,
    div5longestgtime INT,
    div5wheelsoff STRING,
    div5tailnum STRING
)
PARTITIONED BY (year STRING)
STORED AS PARQUET
```

```
LOCATION 's3://athena-examples-myregion/flight/parquet/'
tblproperties ("parquet.compression"="SNAPPY");
```

Exécutez l'instruction `MSCK REPAIR TABLE` sur la table pour actualiser les métadonnées de la partition:

```
MSCK REPAIR TABLE flight_delays_pq;
```

Recherchez les 10 principaux itinéraires dont le retard dépasse 1 heure :

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_pq
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

> **Note**
>
> Les données du tableau des vols proviennent des Vols fournis par le ministère américain des transports, Bureau of Transportation Statistics (Bureau des statistiques des transports). Désaturé de l'original.

Omission des statistiques Parquet

Lorsque vous lisez des données Parquet, vous pouvez recevoir des messages d'erreur tels que les suivants :

```
HIVE_CANNOT_OPEN_SPLIT: Index x out of bounds for length y
HIVE_CURSOR_ERROR: Failed to read x bytes
HIVE_CURSOR_ERROR: FailureException at Malformed input: offset=x
HIVE_CURSOR_ERROR: FailureException at java.io.IOException:
can not read class org.apache.parquet.format.PageHeader: Socket is closed by peer.
```

Pour contourner ce problème, utilisez l'ALTER TABLE SET TBLPROPERTIESinstruction CREATE TABLE or pour définir la SerDe `parquet.ignore.statistics` propriété Parquet sur`true`, comme dans les exemples suivants.

Exemple CREATE TABLE

```
...
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES (
'parquet.ignore.statistics'='true')
STORED AS PARQUET
...
```

Exemple ALTER TABLE

```
ALTER TABLE ... SET TBLPROPERTIES ('parquet.ignore.statistics'='true')
```

Régex SerDe

La Regex SerDe utilise une expression régulière (regex) pour désérialiser les données en extrayant des groupes d'expressions régulières dans des colonnes de table.

Si une ligne des données ne correspond pas à l'expression régulière, toutes les colonnes de la ligne sont renvoyées comme `NULL`. Si une ligne correspond à l'expression rationnelle mais a moins de groupes que prévu, les groupes manquants sont `NULL`. Si une ligne des données correspond à l'expression régulière mais contient plus de colonnes que de groupes dans l'expression régulière, les colonnes supplémentaires sont ignorées.

Pour plus d'informations, consultez Class RegexSerDe dans la documentation d'Apache Hive.

SerDe nom

RegexSerDe

Nom de bibliothèque

RegexSerDe

Exemples

L'exemple suivant crée une table à partir de CloudFront journaux à l'aide du RegExSerDe. Remplacez *myregion* dans `s3://athena-examples-`*myregion*`/cloudfront/plaintext/` par l'identifiant de région dans lequel vous exécutez Athena (par exemple, `s3://athena-examples-us-west-1/cloudfront/plaintext/`).

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_logs (
```

```
  `Date` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
  BrowserVersion STRING
 )
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
 WITH SERDEPROPERTIES (
 "input.regex" = "^(?!#)([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\
\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+[^\(]+[\(]([^\;]+).*\%20([^\/]+)[\/]
(.*)$"
 )
LOCATION 's3://athena-examples-myregion/cloudfront/plaintext/';
```

Exécution de requêtes SQL à l'aide d'Amazon Athena

Vous pouvez exécuter des requêtes SQL à l'aide d'Amazon Athena sur des sources de données enregistrées dans le AWS Glue Data Catalog et des sources de données telles que des métastores Hive et des instances Amazon DocumentDB auxquelles vous vous connectez à l'aide de la fonction de requête fédérée d'Athena. Pour de plus amples informations sur l'utilisation des sources de données, veuillez consulter Connexion aux sources de données. Lorsque vous exécutez une requête DDL (Data Definition Language) qui modifie un schéma, Athena écrit les métadonnées dans le métastore associé à la source de données. En outre, certaines requêtes, telles que `CREATE TABLE AS` et `INSERT INTO`, peuvent écrire des registres dans le jeu de données, par exemple en ajoutant un registre CSV à un emplacement Simple Storage Service (Amazon S3). Lorsque vous exécutez une requête, Athena enregistre les résultats dans un emplacement de résultats de requête que vous spécifiez. Cela vous permet d'afficher l'historique des requêtes, ainsi que de télécharger et d'afficher les jeux de résultats des requêtes.

Cette section fournit des conseils sur l'exécution de requêtes Athena sur des sources de données et des types de données courants à l'aide de diverses instructions SQL. Des conseils généraux sont fournis pour l'utilisation des structures et des opérateurs courants, par exemple pour l'utilisation des

tableaux, la concaténation, le filtrage, l'aplatissement et le tri. D'autres exemples incluent les requêtes de données dans des tables avec des structures et des cartes imbriquées, des tables basées sur des ensembles de données codés en JSON et des ensembles de données associés Services AWS tels que les journaux et les journaux AWS CloudTrail Amazon EMR. Cette documentation n'a pas pour objectif de couvrir en détail l'utilisation du langage SQL standard. Pour plus d'informations sur SQL, veuillez consulter les références de langage de Trino et Presto.

Rubriques

- Affichage des plans d'exécution pour les requêtes SQL
- Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie
- Réutilisation des résultats des requêtes
- Affichage des statistiques et des détails d'exécution pour les requêtes terminées
- Utilisation des vues
- Utilisation de requêtes enregistrées
- Utilisation de requêtes paramétrées
- Utilisation de l'optimiseur basé sur les coûts
- Interrogation des données de S3 Express One Zone
- Interrogation d'objets Amazon S3 Glacier restaurés
- Traitement des mises à jour de schéma
- Interrogation des tableaux
- Interrogation de données géospatiales
- Interrogation de JSON
- Utilisation du Machine Learning (ML) avec Amazon Athena
- Interrogation avec des fonctions définies par l'utilisateur
- Requête entre régions
- Interrogation du AWS Glue Data Catalog
- Journaux d'interrogation Service AWS
- Interrogation des journaux de serveur web stockés dans Simple Storage Service (Amazon S3)

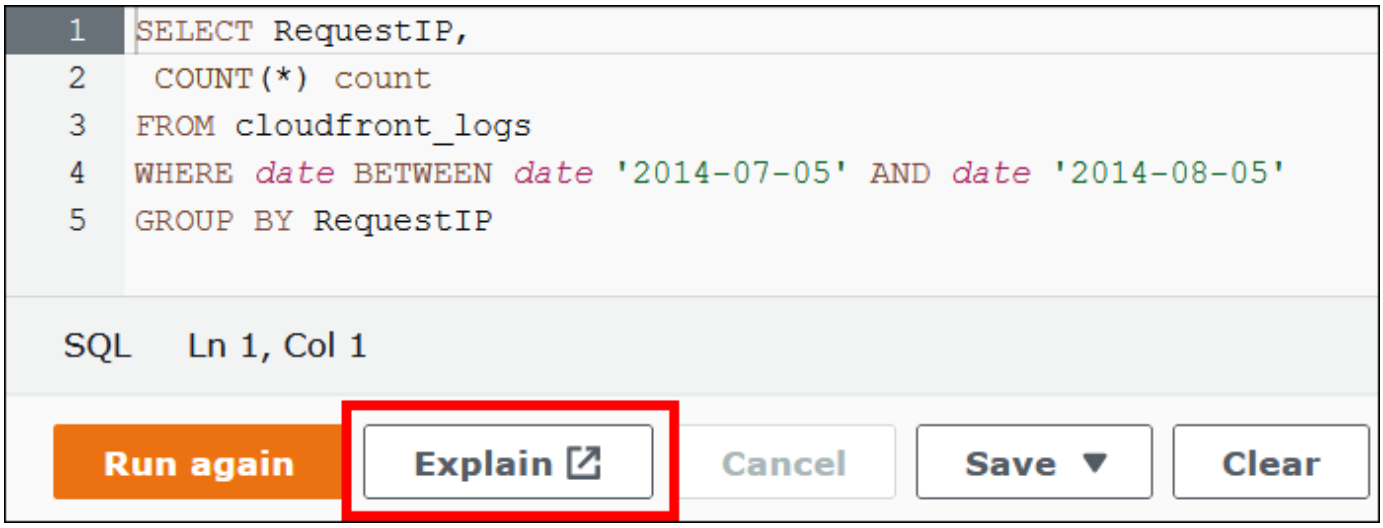
Pour les considérations et les restrictions, consultez Considérations et limitations relatives aux requêtes SQL dans Amazon Athena.

Affichage des plans d'exécution pour les requêtes SQL

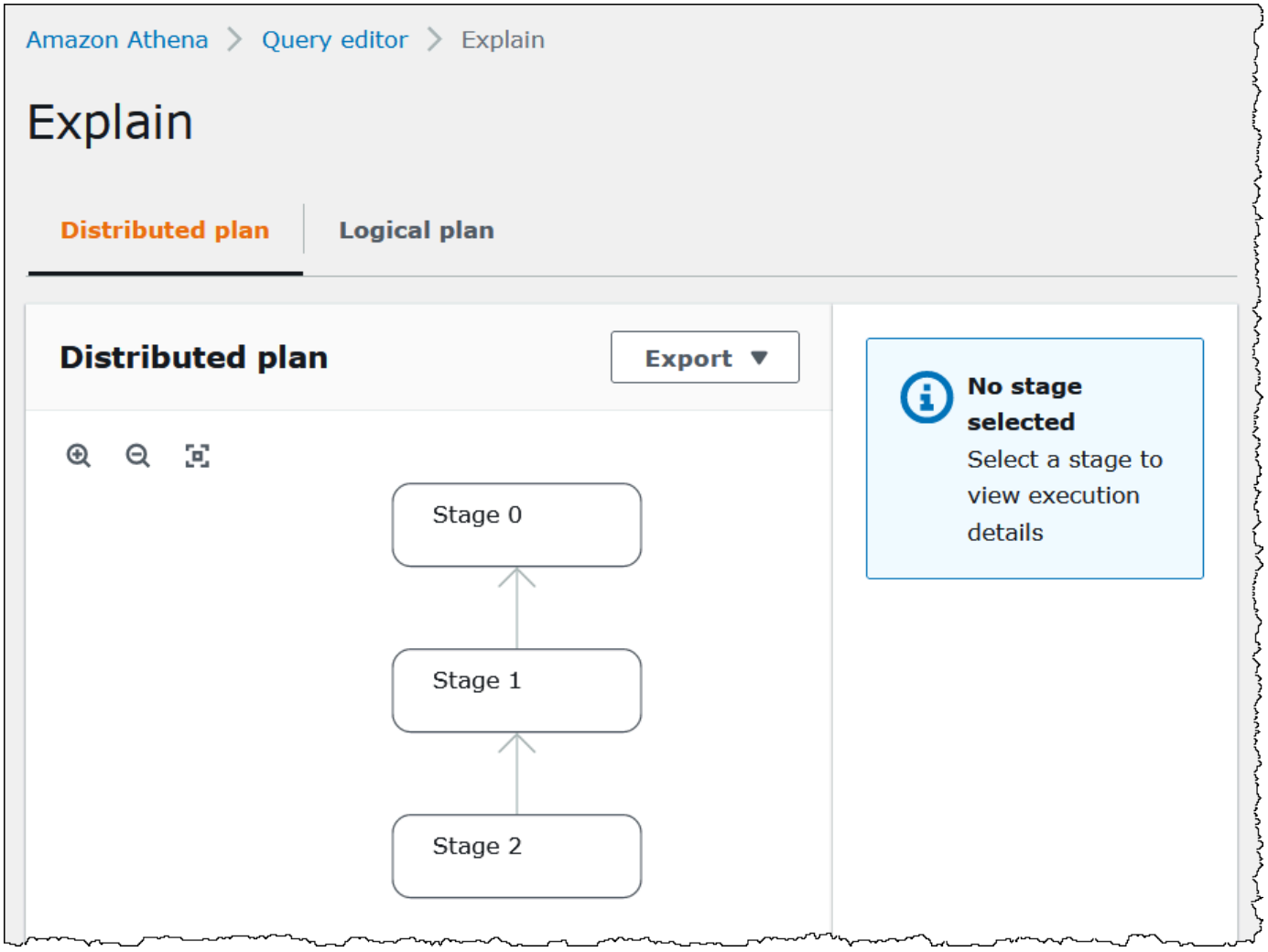
Vous pouvez utiliser l'éditeur de requêtes Athena pour voir des représentations graphiques de la manière dont votre requête sera exécutée. Lorsque vous entrez une requête dans l'éditeur et que vous choisissez l'option EXPLAIN, Athena utilise une instruction SQL EXPLAIN sur votre requête pour créer deux graphiques correspondants : un plan d'exécution distribué et un plan d'exécution logique. Vous pouvez utiliser ces graphiques pour analyser, dépanner et améliorer l'efficacité de vos requêtes.

Pour afficher les plans d'exécution d'une requête

1. Saisissez votre requête dans l'éditeur de requête Athena, puis choisissez Explain.



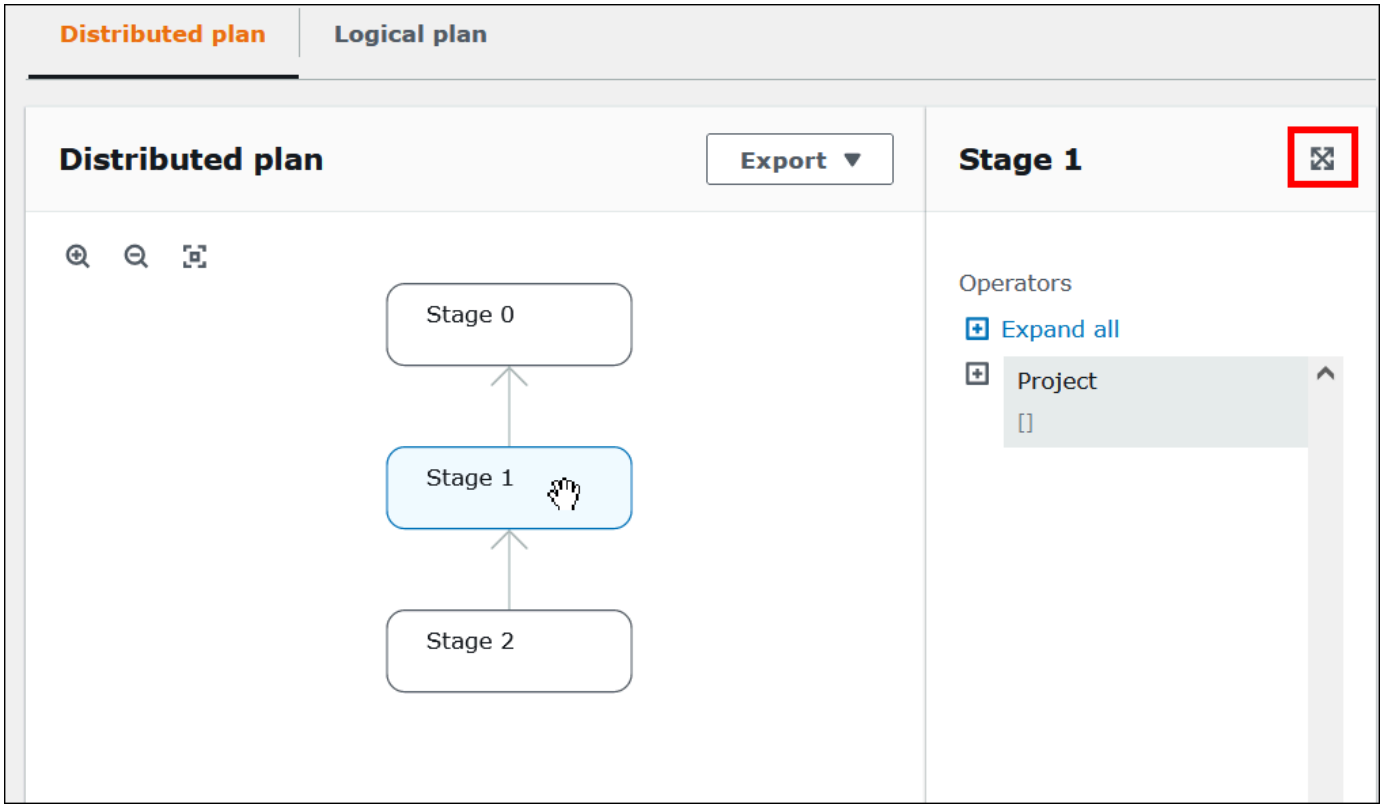
   L'onglet Distributed plan (Plan distribué) affiche le plan d'exécution de votre requête dans un environnement distribué. Un plan distribué comporte des fragments de traitement ou d'étapes. Chaque étape possède un numéro d'index de base zéro et est traitée par un ou plusieurs nœuds. Les données peuvent être échangées entre les nœuds.



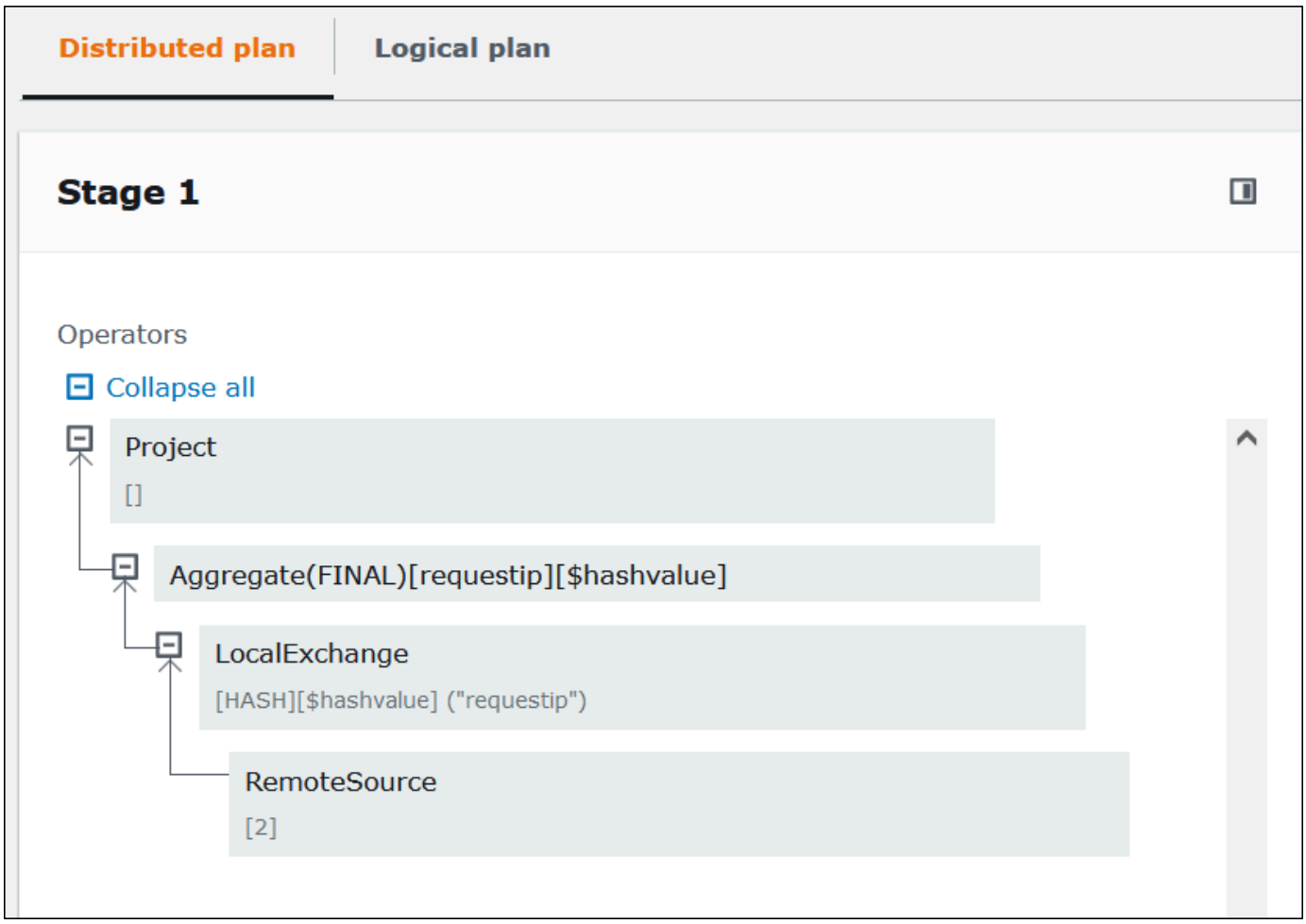
2. Pour parcourir le graphique, utilisez les options suivantes :

   - Pour effectuer un zoom avant ou arrière, faites défiler la souris ou utilisez les icônes de grossissement.
   - Pour ajuster le graphique à l'écran, choisissez l'option Zoom to fit (Zoom sur la taille).
   - Pour déplacer le graphique, maintenez le pointeur de la souris et glissez-le.

3. Pour afficher les détails d'une étape, sélectionnez-la.

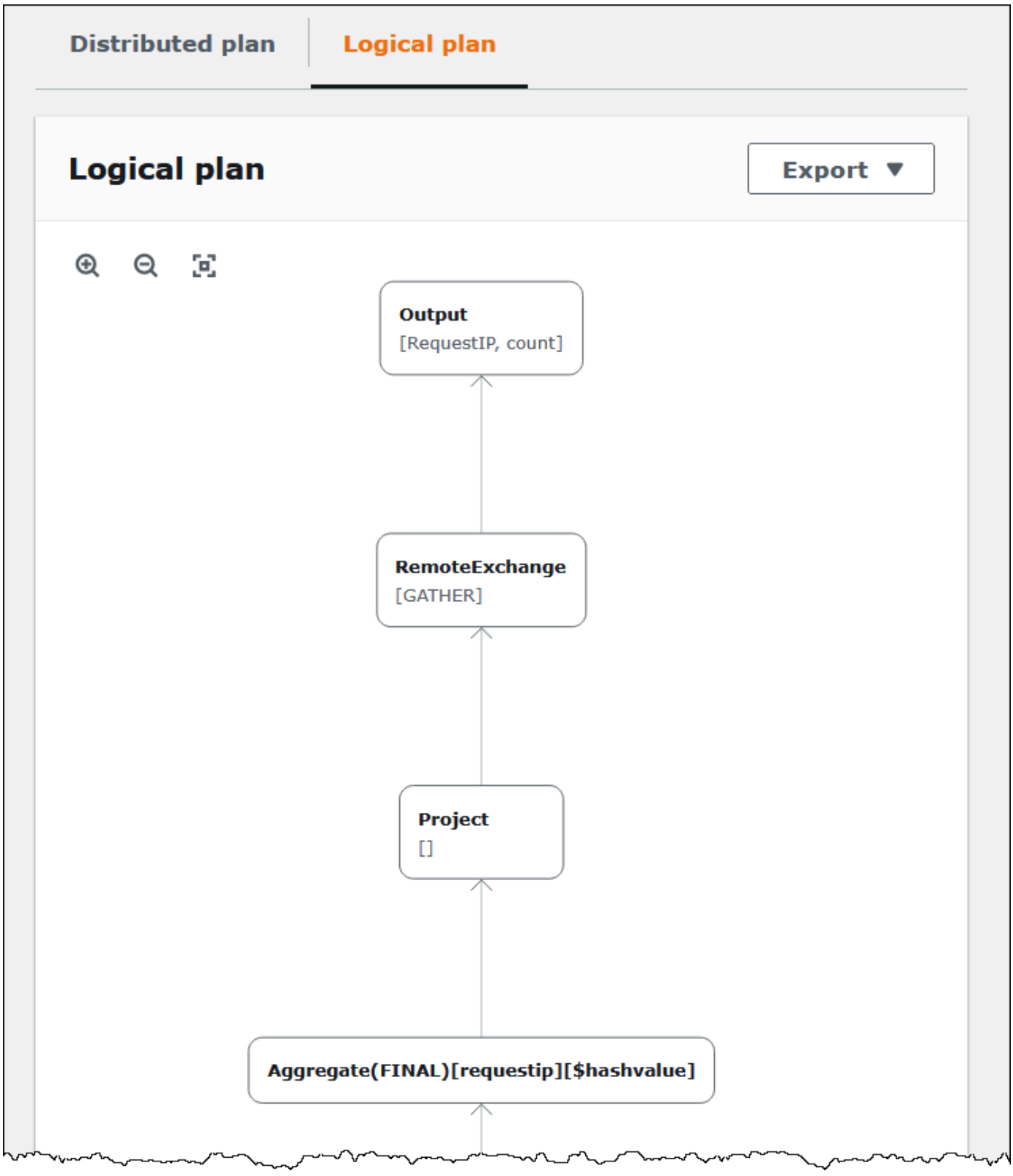


4. Pour voir les détails de l'étape dans toute sa largeur, cliquez sur l'icône de développement en haut à droite du volet des détails.
5. Pour plus de détails, développez un ou plusieurs éléments dans l'arborescence des opérateurs. Pour plus d'informations sur les fragments de plan distribués, consultez Types de sorties de l'instruction EXPLAIN.



> **Important**
>
> Actuellement, certains filtres de partition peuvent ne pas être visibles dans l'arborescence des opérateurs imbriqués même si Athena les applique à votre requête. Pour vérifier l'effet de tels filtres, exécutez EXPLAIN ou EXPLAIN ANALYZE sur votre requête et affichez les résultats.

6. Cliquez sur l'onglet Logical plan (Plan logique). Le graphique montre le plan logique d'exécution de votre requête. Pour plus d'informations sur les termes opérationnels, consultez Explication des résultats de l'instruction EXPLAIN d'Athena.



7. Pour exporter un plan sous forme d'image SVG ou PNG, ou sous forme de texte JSON, choisissez Export (Exporter).

Ressources supplémentaires

Pour plus d'informations, veuillez consulter les ressources suivantes.

Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena

Explication des résultats de l'instruction EXPLAIN d'Athena

Affichage des statistiques et des détails d'exécution pour les requêtes terminées

Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie

Amazon Athena stocke automatiquement les résultats de requête et les informations de métadonnées de chaque requête s'exécutant dans un emplacement de résultats de requête que vous pouvez spécifier dans Simple Storage Service (Amazon S3). Si nécessaire, vous pouvez accéder aux fichiers de cet emplacement pour les utiliser. Vous pouvez également télécharger les fichiers de résultats de requête directement à partir de la console Athena.

Pour configurer un emplacement de résultats de requête Simple Storage Service (Amazon S3) pour la première fois, voir Spécification d'un emplacement de résultats de requête à l'aide de la console Athena.

Les fichiers de sortie sont enregistrés automatiquement pour chaque requête qui s'exécute. Pour accéder aux fichiers de sortie de requête et les visualiser à l'aide de la console Athena, les principaux IAM (utilisateurs et rôles) doivent être autorisés à effectuer l'GetObjectaction Amazon S3 pour l'emplacement des résultats de la requête, ainsi que pour l'action Athena. GetQueryResults L'emplacement des résultats de requête peut être chiffré. Si l'emplacement est chiffré, les utilisateurs doivent disposer des autorisations de clé appropriées pour chiffrer et déchiffrer l'emplacement des résultats de requête.

> **Important**
>
> Les principals IAM disposant de l'autorisation d'utilisation de l'action `GetObject` de Simple Storage Service (Amazon S3) sur l'emplacement des résultats de requête peuvent récupérer les résultats de requête à partir de Simple Storage Service (Amazon S3) même si l'autorisation d'utilisation de l'action `GetQueryResults` d'Athena leur est refusée.

Spécification d'un emplacement de résultats de requête

L'emplacement des résultats de requête qu'Athena utilise est déterminé par une combinaison de paramètres de groupe de travail et de paramètres côté client. Les paramètres côté client sont basés sur la façon dont vous exécutez la requête.

- Si vous exécutez la requête à l'aide de la console Athena, c'est l'emplacement des résultats de la requête saisi sous Settings (Paramètres) dans la barre de navigation qui détermine le paramètre côté client.
- Si vous exécutez la requête à l'aide de l'API Athena, le `OutputLocation` paramètre de l'StartQueryExecutionaction détermine le paramètre côté client.
- Si vous utilisez les pilotes ODBC ou JDBC pour exécuter des requêtes, c'est la propriété `S3OutputLocation` spécifiée dans l'URL de connexion qui détermine les paramètres côté client.

> **Important**
>
> Lorsque vous exécutez une requête à l'aide de l'API ou du pilote ODBC ou JDBC, le paramètre de console ne s'applique pas.

Chaque configuration de groupe de travail possède une option Override client-side settings (Remplacer les paramètres côté)qui peut être activée. Lorsque cette option est activée, les paramètres du groupe de travail ont la priorité sur les paramètres côté client applicables lorsqu'un principal IAM associé à ce groupe de travail exécute la requête.

Spécification d'un emplacement de résultats de requête à l'aide de la console Athena

Pour pouvoir exécuter une requête, vous devez spécifier un emplacement de compartiment de résultats de requête dans Simple Storage Service (Amazon S3) ou utiliser un groupe de travail qui a spécifié un compartiment et dont la configuration remplace les paramètres du client.

Spécification d'un emplacement de résultats de requête de paramètre côté client à l'aide de la console Athena

1. Passez au groupe de travail pour lequel vous voulez spécifier l'emplacement de résultats de requête. Le nom du groupe de travail par défaut est primaire.
2. Dans la barre de navigation, choisissez Settings (Paramètres).
3. Dans la barre de navigation, choisissez Manage (Gérer).

4. Pour Manage settings (Gestion des paramètres), procédez de l'une des manières suivantes :

   - Dans la zone de texte Location of query result (Emplacement des résultats de la requête), saisissez le chemin d'accès au compartiment que vous avez créé dans Simple Storage Service (Amazon S3) pour les résultats de votre requête. Préfixez le chemin avec `s3://`.
   - Choisissez Parcourir S3, choisissez le compartiment Simple Storage Service (Amazon S3) que vous avez créé pour votre région actuelle, puis sélectionnez Choisir.

   > **Note**
   >
   > Si vous utilisez un groupe de travail qui spécifie un emplacement de résultats de requête pour tous les utilisateurs du groupe de travail, l'option permettant de modifier l'emplacement de résultats de requête n'est pas disponible.

5. (Facultatif) Choisissez View lifecycle configuration (Afficher la configuration du cycle de vie) pour afficher et configurer les règles de cycle de vie d'Amazon S3 dans votre compartiment de résultats de requêtes. Les règles de cycle de vie d'Amazon S3 que vous créez peuvent être des règles d'expiration ou des règles de transition. Les règles d'expiration suppriment automatiquement les résultats des requêtes après un certain laps de temps. Les règles de transition les déplacent vers un autre niveau de stockage d'Amazon S3. Pour de plus amples informations, veuillez consulter la section Définition d'une configuration de cycle de vie sur un compartiment dans le Guide de l'utilisateur Amazon Simple Storage Service.
6. (Facultatif) Pour Propriétaire attendu du bucket, entrez l'ID du bucket Compte AWS que vous pensez être le propriétaire du bucket d'emplacement de sortie. Il s'agit d'une mesure de sécurité supplémentaire. Si l'ID de compte du propriétaire du compartiment ne correspond pas à l'ID que vous spécifiez, les tentatives de sortie vers le compartiment échoueront. Pour obtenir des informations détaillées, consultez Vérification de la propriété du compartiment avec la condition de propriétaire du compartiment dans le Guide de l'utilisateur Simple Storage Service (Amazon S3).

   > **Note**
   >
   > Le paramètre de propriétaire du compartiment attendu s'applique uniquement à l'emplacement de sortie Simple Storage Service (Amazon S3) que vous spécifiez pour les résultats de la requête Athena. Il ne s'applique pas aux autres emplacements Simple Storage Service (Amazon S3) tels que les emplacements de source de données dans des compartiments Simple Storage Service (Amazon S3) externes, des emplacements

de table de destination `CTAS` et `INSERT INTO`, des emplacements de sortie d'instruction `UNLOAD`, des opérations de déversement de compartiments pour les requêtes fédérées, ou des requêtes `SELECT` exécutées sur une table d'un autre compte.

7. (Facultatif) Choisissez Encrypt query results (Chiffrer les résultats de requête) si vous souhaitez chiffrer les résultats des requêtes stockées dans Simple Storage Service (Amazon S3). Pour plus d'informations sur le chiffrement EBS dans Athena, veuillez consulter Chiffrement au repos.
8. (Facultatif) Choisissez Assign bucket owner full control over query results (Attribuer au propriétaire du compartiment un contrôle total sur les résultats) pour accorder un contrôle total sur les résultats de la requête au propriétaire du compartiment lorsque les listes de contrôle d'accès (ACL) sont activées pour le compartiment de résultats de la requête. Par exemple, si l'emplacement de résultat de votre requête appartient à un autre compte, vous pouvez accorder la propriété et le contrôle total des résultats de vos requêtes à l'autre compte. Pour plus d'informations, veuillez consulter la section Contrôle de la propriété des objets et désactivation des ACL pour votre compartiment dans le Guide de l'utilisateur d'Simple Storage Service (Amazon S3).
9. Choisissez Enregistrer.

Emplacements par défaut précédemment créés

Auparavant, sur Athena, si vous exécutiez une requête sans spécifier de valeur pour Query result location (Emplacement de résultats de requête), et que le paramètre d'emplacement de résultats de requête n'était pas remplacé par un groupe de travail, Athena créait un emplacement par défaut. L'emplacement par défaut était`aws-athena-query-results-`*`MyAcctID`*`-`*`MyRegion`*, où *`MyAcctID`* était l'ID de compte Amazon Web Services du principal IAM qui a exécuté la requête, et *`MyRegion`*était la région où la requête a été exécutée (par exemple,`us-west-1`.)

Maintenant, pour pouvoir exécuter une requête Athena dans une région dans laquelle votre compte n'a pas utilisé Athena précédemment, vous devez spécifier un emplacement de résultats de requête ou utiliser un groupe de travail qui remplace le paramètre d'emplacement de résultats de requête. Bien qu'Athena ne crée plus d'emplacement de résultats de requête par défaut pour vous, les emplacements `aws-athena-query-results-`*`MyAcctID`*`-`*`MyRegion`* par défaut créés précédemment restent valides et vous pouvez continuer à les utiliser.

Spécification d'un emplacement de résultats de requête à l'aide d'un groupe de travail

Vous spécifiez l'emplacement de résultats de requête dans une configuration de groupe de travail à l'aide de la AWS Management Console, de la AWS CLI ou de l'API Athena.

Lorsque vous utilisez le AWS CLI, spécifiez l'emplacement du résultat de la requête à l'aide du `OutputLocation` paramètre de l'`--configuration`option lorsque vous exécutez la `aws athena update-work-group`commande `aws athena create-work-group`ou.

Spécification de l'emplacement des résultats de requête pour un groupe de travail à l'aide de la console Athena

1. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

2. Dans le panneau de navigation, choisissez Workgroups (Groupes de travail).
3. Dans la liste des groupes de travail, choisissez le lien du groupe de travail que vous voulez modifier.
4. Choisissez Modifier.
5. Pour l'emplacement et le chiffrement des résultats de requête, effectuez l'une des opérations suivantes :

   - Dans la zone de texte Query result location (Emplacement des résultats de la requête), saisissez le chemin d'accès au compartiment que vous avez créé dans Simple Storage Service (Amazon S3) pour les résultats de votre requête. Préfixez le chemin avec `s3://`.
   - Choisissez Browse S3 (Parcourir S3), choisissez le compartiment Simple Storage Service (Amazon S3) de votre région actuelle que vous souhaitez utiliser, puis choisissez Choose (Choisir).

6. (Facultatif) Pour Propriétaire attendu du bucket, entrez l'ID du bucket Compte AWS que vous pensez être le propriétaire du bucket d'emplacement de sortie. Il s'agit d'une mesure de sécurité

supplémentaire. Si l'ID de compte du propriétaire du compartiment ne correspond pas à l'ID que vous spécifiez, les tentatives de sortie vers le compartiment échoueront. Pour obtenir des informations détaillées, consultez Vérification de la propriété du compartiment avec la condition de propriétaire du compartiment dans le Guide de l'utilisateur Simple Storage Service (Amazon S3).

> **Note**
>
> Le paramètre de propriétaire du compartiment attendu s'applique uniquement à l'emplacement de sortie Simple Storage Service (Amazon S3) que vous spécifiez pour les résultats de la requête Athena. Il ne s'applique pas aux autres emplacements Simple Storage Service (Amazon S3) tels que les emplacements de source de données dans des compartiments Simple Storage Service (Amazon S3) externes, des emplacements de table de destination `CTAS` et `INSERT INTO`, des emplacements de sortie d'instruction `UNLOAD`, des opérations de déversement de compartiments pour les requêtes fédérées, ou des requêtes `SELECT` exécutées sur une table d'un autre compte.

7. (Facultatif) Choisissez Encrypt query results (Chiffrer les résultats de requête) si vous souhaitez chiffrer les résultats des requêtes stockées dans Simple Storage Service (Amazon S3). Pour plus d'informations sur le chiffrement EBS dans Athena, veuillez consulter Chiffrement au repos.
8. (Facultatif) Choisissez Assign bucket owner full control over query results (Attribuer au propriétaire du compartiment un contrôle total sur les résultats) pour accorder un contrôle total sur les résultats de la requête au propriétaire du compartiment lorsque les listes de contrôle d'accès (ACL) sont activées pour le compartiment de résultats de la requête. Par exemple, si l'emplacement de résultat de votre requête appartient à un autre compte, vous pouvez accorder la propriété et le contrôle total des résultats de vos requêtes à l'autre compte.

   Si le paramètre S3 Object Ownership du compartiment est défini à Propriétaire du compartiment préféré, le propriétaire du compartiment possède également tous les objets de résultats de requête écrits à partir de ce groupe de travail. Par exemple, si le groupe de travail d'un compte externe active cette option et définit son emplacement de résultat de requête sur le compartiment Simple Storage Service (Amazon S3) de votre compte qui dispose d'un paramètre de S3 Object Ownership dont la valeur est définie à Propriétaire du compartiment préféré, vous possédez et contrôlez complètement les résultats de requête du groupe de travail externe.

   Sélectionner cette option lorsque le paramètre S3 Object Ownership du compartiment de résultats de requête est défini à Propriétaire du compartiment appliqué n'a aucun effet. Pour plus

d'informations, veuillez consulter la section Contrôle de la propriété des objets et désactivation des ACL pour votre compartiment dans le Guide de l'utilisateur d'Simple Storage Service (Amazon S3).

9. Si vous souhaitez demander à tous les utilisateurs du groupe de travail d'utiliser l'emplacement des résultats de la requête que vous avez spécifié, faites défiler l'écran jusqu'à la section Settings (Paramètres) et sélectionnez Override client-side settings (Remplacer les paramètres côté client).
10. Sélectionnez Enregistrer les modifications.

Téléchargement des fichiers de résultats de requête à l'aide de la console Athena

Vous pouvez télécharger le fichier CSV des résultats de la requête à partir du volet de la requête immédiatement après avoir exécuté une requête. Vous pouvez également télécharger les résultats de requête à partir de requêtes récentes à partir de l'onglet Recent queries (Requêtes récentes).

> **Note**
>
> Les fichiers de résultats de requêtes Athena sont des fichiers de données contenant des informations qui peuvent être configurées par des utilisateurs individuels. Certains programmes qui lisent et analysent ces données peuvent potentiellement interpréter certaines de ces données comme des commandes (injection CSV). C'est pourquoi, lorsque vous importez des données CSV de résultats de requêtes dans un tableur, ce dernier peut vous mettre en garde contre des problèmes de sécurité. Pour assurer la sécurité de votre système, vous devez toujours choisir de désactiver les liens ou les macros dans les résultats de requêtes téléchargés.

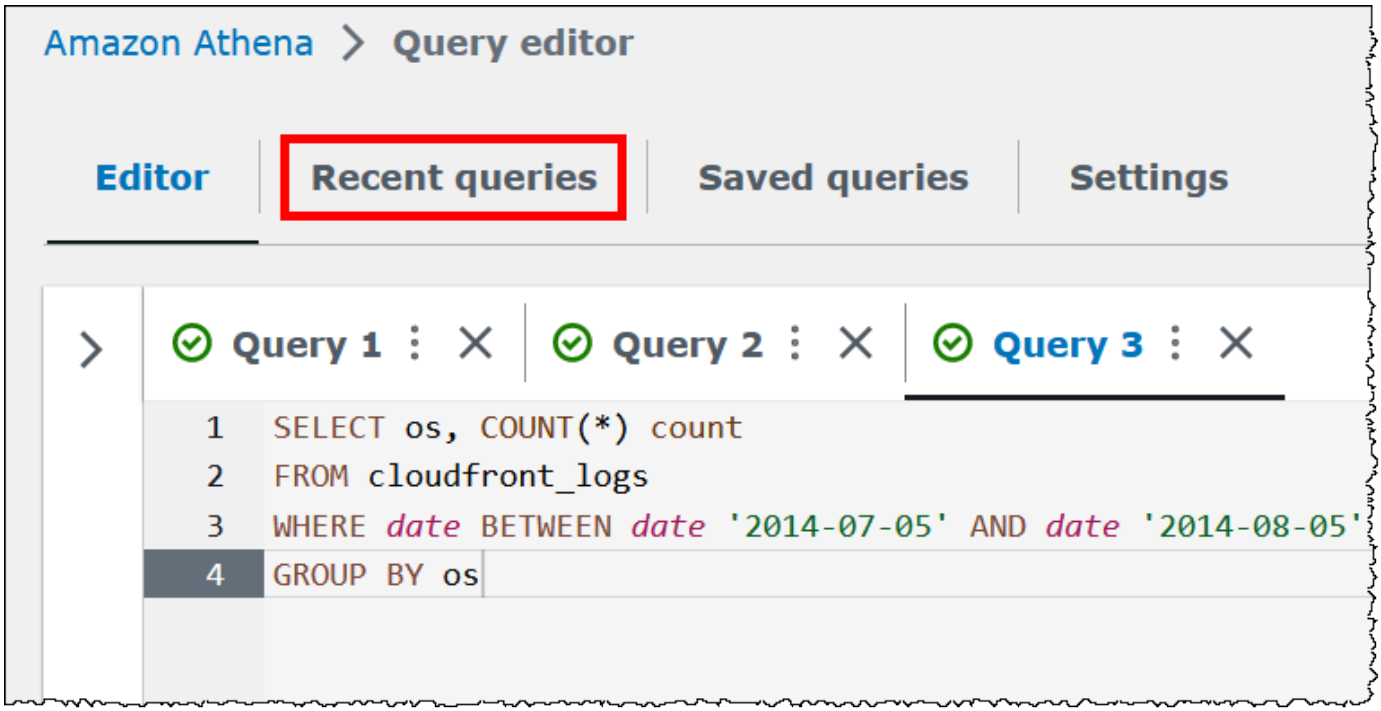
Pour exécuter une requête et télécharger les résultats de requête

1. Saisissez votre requête dans l'éditeur de requête, puis choisissez Run query (Exécuter la requête).

   Une fois l'exécution de la requête terminée, le volet Results (Résultats) affiche ses résultats.
2. Pour télécharger un fichier CSV des résultats de requête, choisissez Download results (Télécharger les résultats) au-dessus du volet des résultats de la requête. En fonction de votre navigateur et de sa configuration, vous devrez peut-être confirmer le téléchargement.

Pour télécharger le fichier des résultats d'une requête antérieure

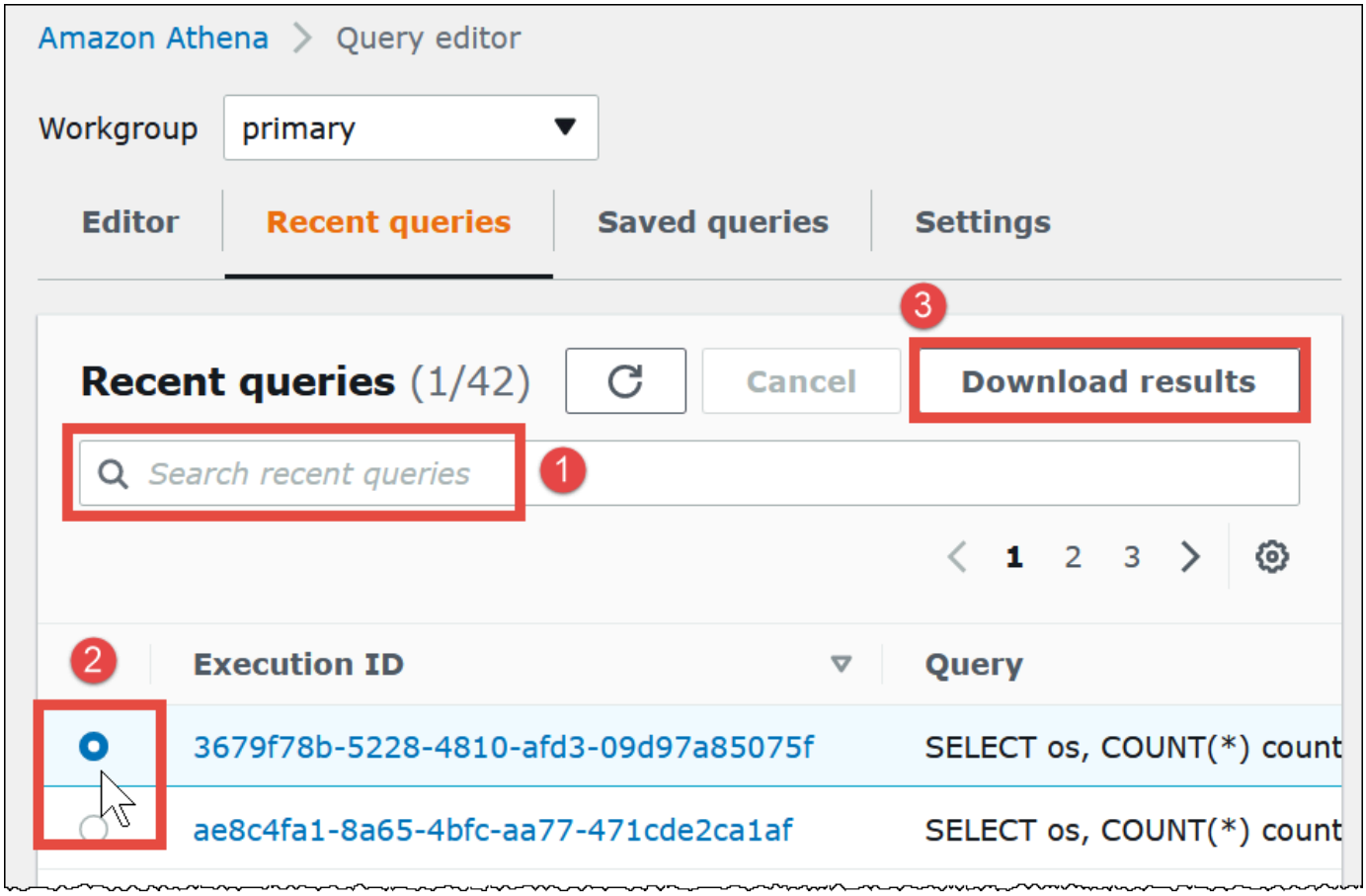
1. Choisissez Recent queries (Requêtes récentes).



2. Utilisez la zone de recherche pour trouver la requête, sélectionnez la requête, puis choisissez Download results (Télécharger les résultats).

> **Note**
>
> Vous ne pouvez pas utiliser l'option Download results (Télécharger les résultats) pour récupérer des résultats de requête qui ont été supprimés manuellement ni pour

récupérer des résultats de requête qui ont été supprimés ou déplacés vers un autre emplacement conformément aux règles de cycle de vie d'Amazon S3.



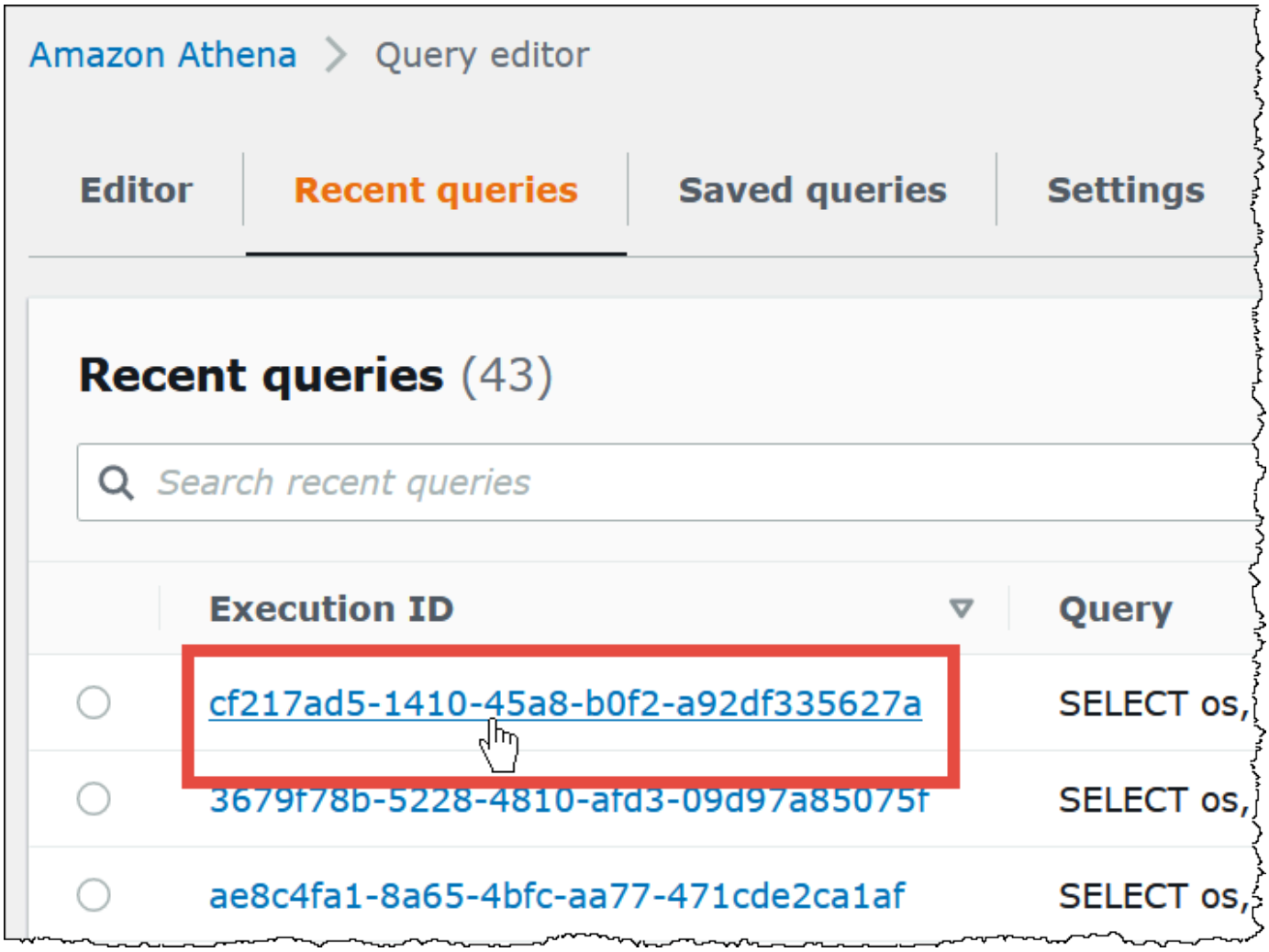
Affichage des requêtes récentes

Vous pouvez utiliser la console Athena pour voir quelles requêtes ont réussi ou échoué, et afficher les détails des erreurs pour les requêtes ayant échoué. Athena conserve un historique des requêtes pendant 45 jours.

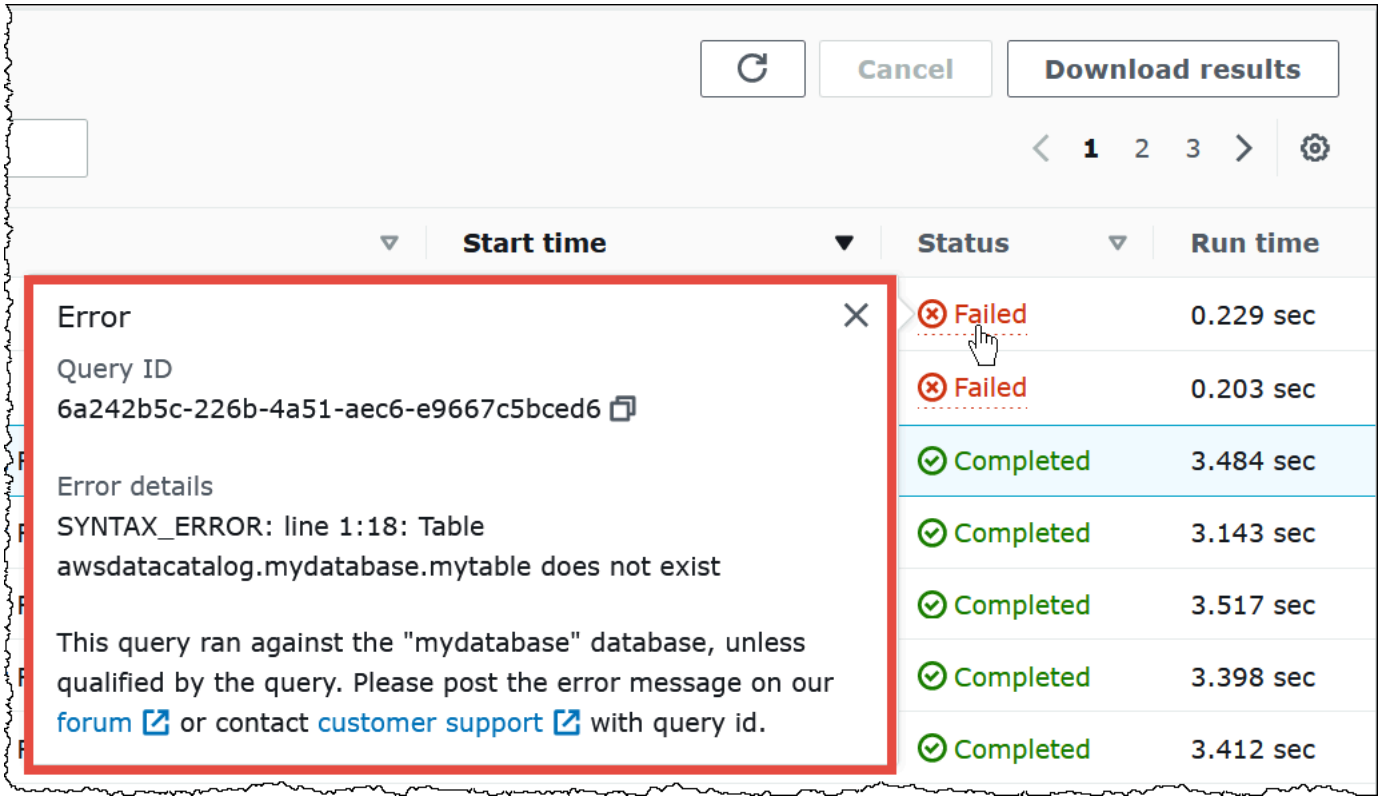
Pour afficher les résultats de requête dans la console Athena

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Choisissez Recent queries (Requêtes récentes). L'onglet Recent queries (Requêtes récentes) affiche des informations sur chaque requête exécutée.

3. Pour ouvrir une instruction de requête dans l'éditeur de requêtes, choisissez l'ID d'exécution de la requête.



4. Pour afficher les détails d'une requête qui a échoué, choisissez le lien Failed (Échec) pour la requête.

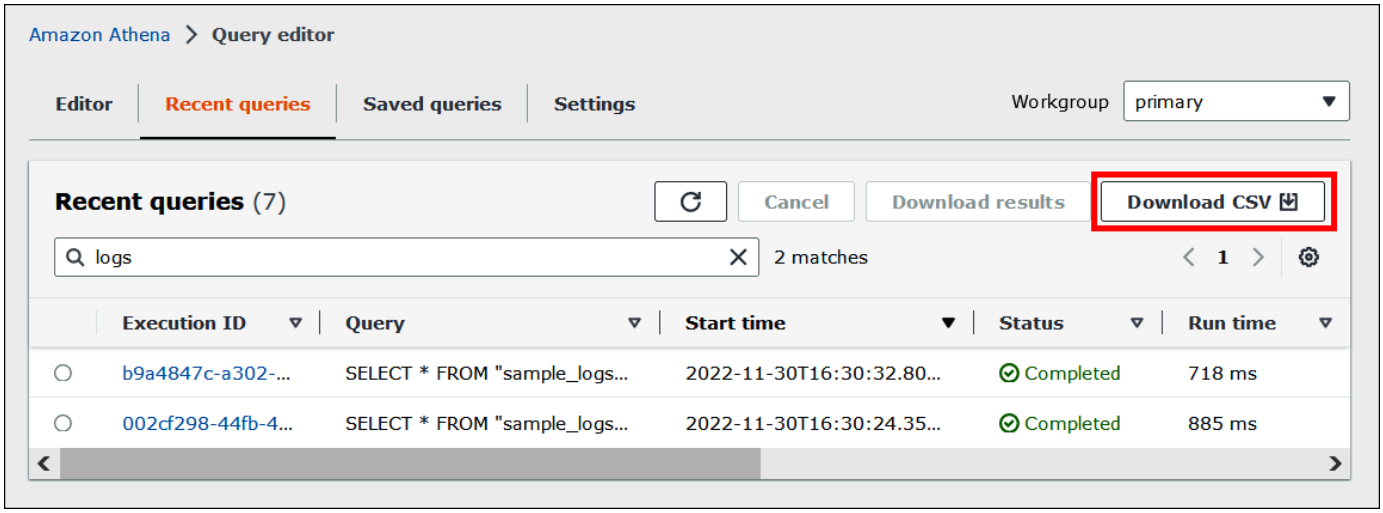


Téléchargement de plusieurs requêtes récentes vers un fichier CSV

Vous pouvez utiliser l'onglet Recent queries (Requêtes récentes) de la console Athena pour exporter une ou plusieurs requêtes récentes vers un fichier CSV afin de les afficher sous forme de table. Le fichier téléchargé ne contient pas les résultats de la requête, mais la chaîne de requête SQL elle-même et d'autres informations sur la requête. Les champs exportés comprennent l'ID de l'exécution, le contenu de la chaîne de requête, l'heure de début de la requête, l'état, la durée d'exécution, la quantité de données analysées, la version du moteur de requête utilisée et la méthode de chiffrement. Vous pouvez exporter un maximum de 500 requêtes récentes ou un maximum de 500 requêtes filtrées selon les critères que vous avez saisis dans le champ de recherche.

Exporter une ou plusieurs requêtes récentes vers un fichier CSV

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Choisissez Recent queries (Requêtes récentes).
3. (Facultatif) Utilisez le champ de recherche pour filtrer les dernières requêtes à télécharger.
4. Choisissez Télécharger le rapport CSV.



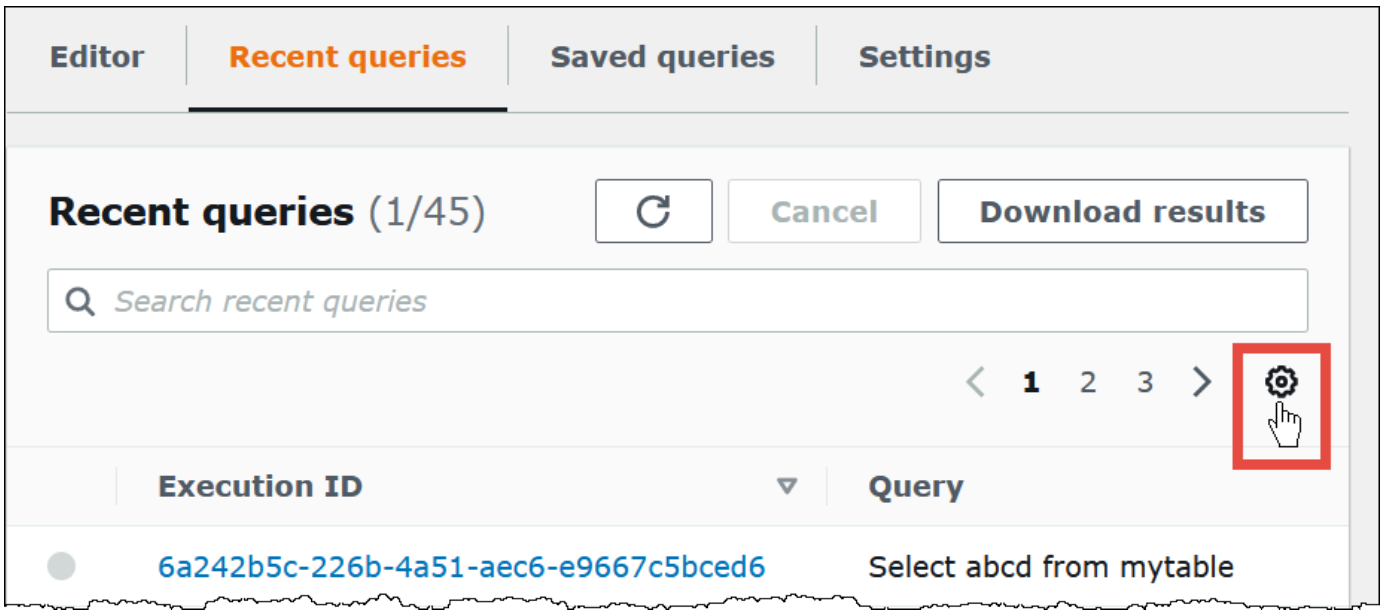
5. À l'invite d'enregistrement du fichier, choisissez Save (Enregistrer). Le nom de fichier par défaut est `Recent Queries` suivi d'un horodatage (par exemple, `Recent Queries 2022-12-05T16 04 27.352-08 00.csv`)

Configuration des options d'affichage des requêtes récentes

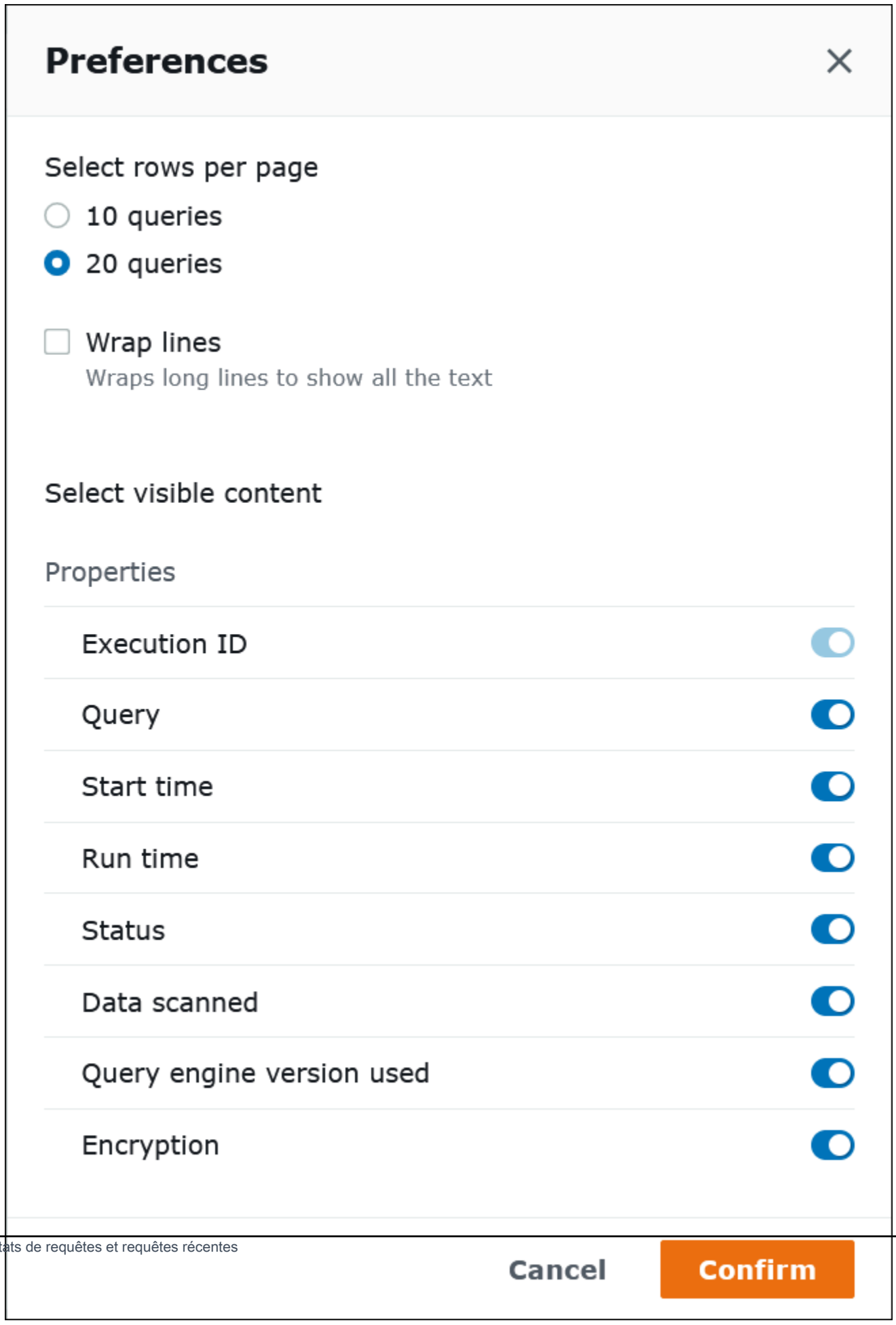
Vous pouvez configurer des options pour l'onglet Requêtes récentes (Requêtes récentes), comme les colonnes à afficher et l'habillage du texte.

Configurer les options de l'onglet Recent queries (Requêtes récentes)

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Choisissez Recent queries (Requêtes récentes).
3. Cliquez sur le bouton des options (icône en forme d'engrenage).



4. Dans la boite de dialogue Preferences (Préférences), choisissez le nombre de lignes par page, le comportement d'encapsulation des lignes et les colonnes à afficher.


Preferences
Select rows per page
10 queries
20 queries
Wrap lines
Wraps long lines to show all the text
Select visible content
Properties
Execution ID
Query
Start time
Run time
Status
Data scanned
Query engine version used
Encryption
Cancel
Confirm

5. Choisissez Confirm (Confirmer).

Conservation de l'historique des requêtes pendant plus de 45 jours

Si vous souhaitez conserver l'historique des requêtes au-delà de 45 jours, vous pouvez récupérer l'historique des requêtes et l'enregistrer dans un magasin de données tel que Simple Storage Service (Amazon S3). Pour automatiser ce processus, vous pouvez utiliser les actions d'API Athena et Simple Storage Service (Amazon S3) et les commandes CLI. La procédure suivante récapitule ces étapes.

Pour récupérer et enregistrer l'historique des requêtes par programmation

1. Utilisez l'action de ListQueryExecutionsl'API Athena ou la commande list-query-executionsCLI pour récupérer les ID de requête.
2. Utilisez l'action de l'GetQueryExecutionAPI Athena ou la commande get-query-executionCLI pour récupérer des informations sur chaque requête en fonction de son identifiant.
3. Utilisez l'action de l'PutObjectAPI Amazon S3 ou la commande de la CLI put-object pour enregistrer les informations dans Amazon S3.

Recherche de fichiers de sortie de requête dans Simple Storage Service (Amazon S3)

Les fichiers de sortie des requêtes sont stockés dans des sous-dossiers sur Simple Storage Service (Amazon S3) selon le modèle de chemin d'accès suivant, à moins que la requête ne soit effectuée dans un groupe de travail dont la configuration prévaut sur les paramètres côté client. Lorsque la configuration du groupe de travail remplace les paramètres côté client, la requête utilise le chemin d'accès aux résultats spécifié par le groupe de travail.

```
QueryResultsLocationInS3/[QueryName|Unsaved/yyyy/mm/dd/]
```

- *QueryResultsLocationInS3* est l'emplacement du résultat de la requête spécifié soit par les paramètres du groupe de travail, soit par les paramètres côté client. Pour plus d'informations, consultez the section called “Spécification d'un emplacement de résultats de requête” dans la suite de ce document.
- Les sous-dossiers suivants sont créés uniquement pour les requêtes exécutées à partir de la console dont le chemin d'accès aux résultats n'a pas été remplacé par la configuration du groupe de travail. *Les requêtes exécutées depuis AWS CLI ou à l'aide de l'API Athena sont enregistrées directement dans le QueryResultsLocationIn S3.*

- *QueryName*est le nom de la requête pour laquelle les résultats sont enregistrés. Si la requête a été exécutée mais n'a pas été enregistrée, `Unsaved` est utilisé.
- *aaaa/mm/jj* correspond à la date d'exécution de la requête.

Les fichiers associés à une requête `CREATE TABLE AS SELECT` sont stockés dans un sous-dossier `tables`du modèle ci-dessus.

Identification des fichiers de sortie de requête

Les fichiers sont enregistrés dans l'emplacement des résultats de requête dans Simple Storage Service (Amazon S3) en fonction du nom de la requête, de l'ID de cette dernière et de la date à laquelle elle a été exécutée. Les fichiers de chaque requête sont nommés à l'aide d'un *QueryID*, qui est un identifiant unique qu'Athena attribue à chaque requête lors de son exécution.

Les types de fichiers suivants sont enregistrés :

Type de fichier	Modèles d'affectation de nom aux fichiers	Description
Fichiers de résultats de requête	*QueryID*.csv *QueryID*.txt	Les fichiers de résultats de requête DML sont enregistrés au format CSV (valeurs séparées par des virgules). Les résultats de requête DDL sont enregistrés sous forme de fichiers texte brut. Vous pouvez télécharger les fichiers de résultats de la console à partir du Results (Résultats) lorsque vous utilisez la console ou à partir de History (Historique) des requêtes. Pour plus d'informations, consultez Téléchargement des fichiers de résultats

Type de fichier	Modèles d'affectation de nom aux fichiers	Description
		de requête à l'aide de la console Athena.
Fichiers de métadonnées de requête	*QueryID*.csv.metadata *QueryID*.txt.metadata	Les fichiers de métadonnées de requête DML et DDL sont enregistrés au format binaire et ne sont pas lisibles par l'utilisateur. L'extension du fichier correspond au fichier des résultats de la requête. Athena utilise les métadonnées lors de la lecture des résultats des requêtes à l'aide de l'action `GetQueryResults` . Bien que ces fichiers puissent être supprimés, nous ne le recommandons pas, car des informations importantes sur la requête sont alors perdues.

Type de fichier	Modèles d'affectation de nom aux fichiers	Description
Fichiers manifestes de données	*QueryID*-manifest.csv	Les fichiers manifeste de données sont générés pour suivre les fichiers qu'Athena crée dans les emplacements des sources de données Simple Storage Service (Amazon S3) lorsqu'une requête INSERT INTO est exécutée. Si une requête échoue, le manifeste suit également les fichiers que la requête avait l'intention d'écrire. Le manifeste est utile pour identifier les fichiers orphelins résultant d'un échec de requête.

Utilisation du AWS CLI pour identifier l'emplacement de sortie de la requête et les fichiers

Pour utiliser le AWS CLI afin d'identifier l'emplacement de sortie de la requête et les fichiers de résultats, exécutez la `aws athena get-query-execution` commande, comme dans l'exemple suivant. Remplacez *abc1234d-5efg-67hi-jklm-89n0op12qr34* par l'ID de requête.

```
aws athena get-query-execution --query-execution-id abc1234d-5efg-67hi-
jklm-89n0op12qr34
```

La commande renvoie un résultat semblable à ce qui suit. Pour une description de chaque paramètre de sortie, reportez-vous get-query-executionà la référence des AWS CLI commandes.

```
{
    "QueryExecution": {
        "Status": {
            "SubmissionDateTime": 1565649050.175,
            "State": "SUCCEEDED",
            "CompletionDateTime": 1565649056.6229999
```

```
        },
        "Statistics": {
            "DataScannedInBytes": 5944497,
            "DataManifestLocation": "s3://DOC-EXAMPLE-BUCKET/athena-query-
results-123456789012-us-west-1/MyInsertQuery/2019/08/12/abc1234d-5efg-67hi-
jklm-89n0op12qr34-manifest.csv",
            "EngineExecutionTimeInMillis": 5209
        },
        "ResultConfiguration": {
            "EncryptionConfiguration": {
                "EncryptionOption": "SSE_S3"
            },
            "OutputLocation": "s3://DOC-EXAMPLE-BUCKET/athena-query-
results-123456789012-us-west-1/MyInsertQuery/2019/08/12/abc1234d-5efg-67hi-
jklm-89n0op12qr34"
        },
        "QueryExecutionId": "abc1234d-5efg-67hi-jklm-89n0op12qr34",
        "QueryExecutionContext": {},
        "Query": "INSERT INTO mydb.elb_log_backup SELECT * FROM mydb.elb_logs LIMIT
 100",
        "StatementType": "DML",
        "WorkGroup": "primary"
    }
}
```

Réutilisation des résultats des requêtes

Lorsque vous réexécutez une requête dans Athena, vous pouvez choisir de réutiliser le dernier résultat stocké de la requête. Cette option peut augmenter les performances et réduire les coûts en termes de nombre d'octets analysés. La réutilisation des résultats des requêtes est utile si, par exemple, vous savez que les résultats ne changeront pas dans un délai donné. Vous pouvez spécifier un âge maximum pour la réutilisation des résultats des requêtes. Athena utilise le résultat stocké tant qu'il n'est pas plus ancien que l'âge que vous avez spécifié. Pour plus d'informations, consultez Réduction des coûts et amélioration des performances des requêtes avec Amazon Athena sur le blog AWS Big Data.

> **Note**
>
> La fonction de réutilisation des résultats des requêtes nécessite la version 3 du moteur Athena. Pour plus d'informations sur le changement de version du moteur Athena, voir Modification des versions du moteur Athena.

Fonctions principales

- La réutilisation des résultats des requêtes est une fonction optionnelle qui s'applique individuellement à chaque requête. Vous pouvez activer la réutilisation des résultats des requêtes pour chaque requête.
- L'âge maximum pour la réutilisation des résultats des requêtes peut être spécifié en minutes, heures ou jours. L'âge maximum pouvant être spécifié est l'équivalent de 7 jours, quelle que soit l'unité de temps utilisée. La valeur par défaut de est de 60 minutes.
- Lorsque vous activez la réutilisation des résultats d'une requête, Athena recherche une exécution précédente de la requête au sein du même groupe de travail. Si le service Athena trouve des résultats de requête stockés correspondants, il ne réexécute pas la requête, mais pointe vers l'emplacement du résultat précédent ou récupère des données à partir de celui-ci.

- Pour toute requête qui active l'option de réutilisation des résultats, Athena réutilise le dernier résultat de la requête enregistré dans le dossier du groupe de travail uniquement lorsque toutes les conditions suivantes sont réunies :
  - La chaîne de requête correspond exactement.
  - Le nom de la base de données et le nom du catalogue correspondent.
  - Le résultat précédent n'est pas plus ancien que l'âge maximum spécifié, ou pas plus ancien que 60 minutes si un âge maximum n'a pas été spécifié.
  - Athena ne réutilise qu'une exécution qui a exactement la même configuration de résultats que l'exécution actuelle.
  - Vous avez accès à toutes les tables référencées dans la requête.
  - Vous avez accès à l'emplacement du fichier S3 où le résultat précédent est stocké.

Si l'une de ces conditions n'est pas remplie, Athena exécute la requête sans utiliser les résultats mis en cache.

Considérations et restrictions

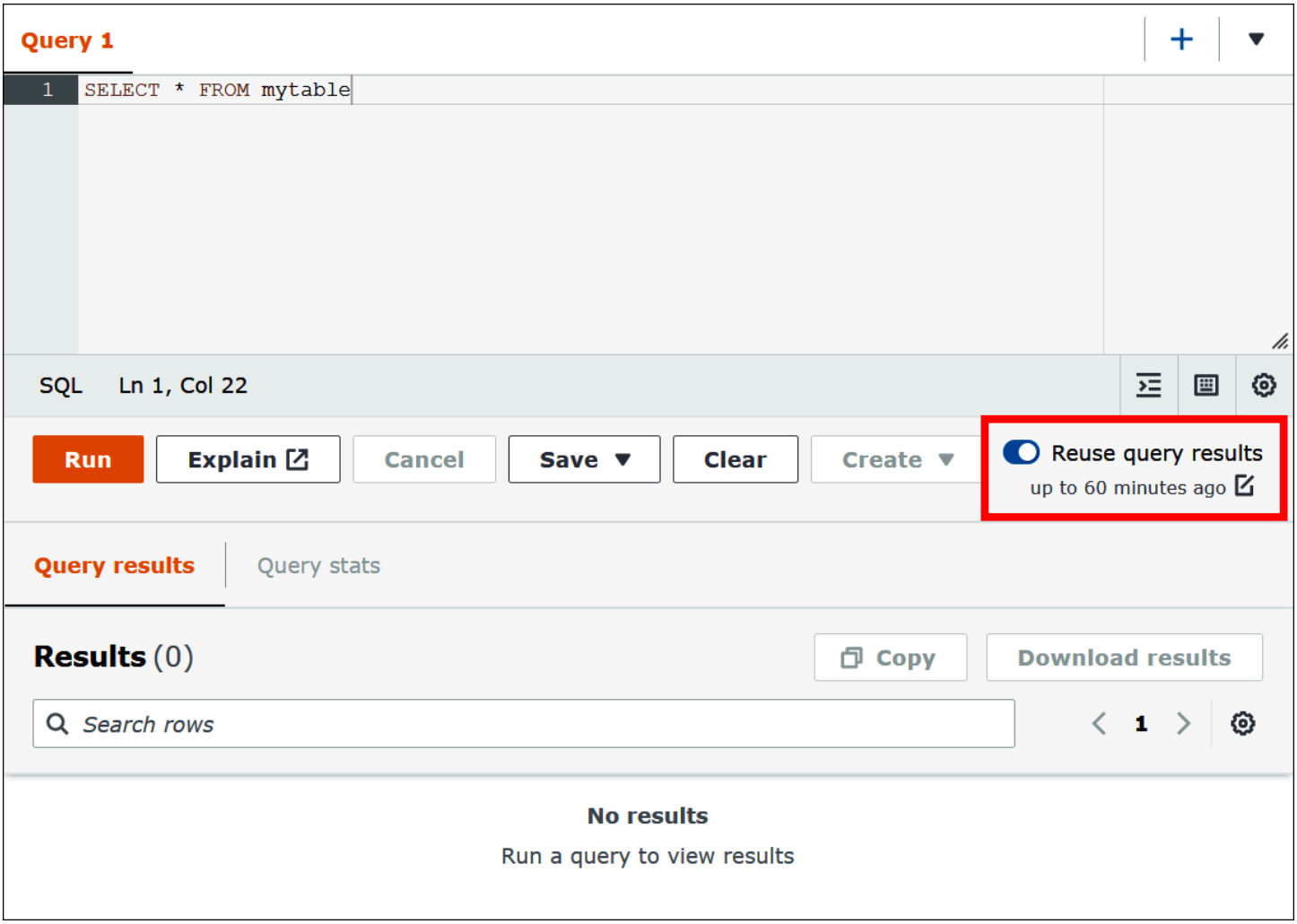
Lorsque vous utilisez la fonction de réutilisation des résultats des requêtes, gardez à l'esprit les points suivants :

- Athena réutilise les résultats des requêtes uniquement au sein du même groupe de travail.
- La fonctionnalité de réutilisation des résultats des requêtes respecte les configurations des groupes de travail. Si vous remplacez la configuration des résultats d'une requête, la fonctionnalité est désactivée.
- Les tables Apache Hive, Apache Hudi, Apache Iceberg et Linux Foundation Delta Lake enregistrées auprès AWS Glue de ce site sont prises en charge. Les métastores Hive externes ne sont pas pris en charge.
- Les requêtes qui font référence à des catalogues fédérés ou à un métastore Hive externe ne sont pas prises en charge.
- La réutilisation des résultats des requêtes n'est pas prise en charge pour les tables régies par Lake Formation.
- La réutilisation des résultats de requête n'est pas prise en charge lorsque l'emplacement Amazon S3 de la source de table est enregistré en tant qu'emplacement de données dans Lake Formation.
- Les tableaux avec des autorisations de lignes et de colonnes ne sont pas pris en charge.
- Les tables dotées d'un contrôle d'accès précis (par exemple, le filtrage des colonnes ou des lignes) ne sont pas prises en charge.
- Toute requête faisant référence à une table non prise en charge n'est pas éligible à la réutilisation des résultats de la requête.
- Athena exige que vous disposiez d'autorisations de lecture sur Amazon S3 pour que le fichier de sortie généré précédemment puisse être réutilisé.
- La fonctionnalité de réutilisation des résultats des requêtes suppose que le contenu du résultat précédent n'a pas été modifié. Athena ne vérifie pas l'intégrité d'un résultat précédent avant de l'utiliser.
- Si les résultats de la requête de l'exécution précédente ont été supprimés ou déplacés vers un autre emplacement dans Amazon S3, l'exécution ultérieure de la même requête ne réutilisera pas les résultats de la requête.
- Des résultats potentiellement périmés peuvent être renvoyés. Athena ne vérifie pas les modifications apportées aux données sources tant que l'âge maximal de réutilisation que vous spécifiez n'est pas atteint.

- Si plusieurs résultats sont disponibles pour la réutilisation, Athena utilise le dernier résultat.
- Les requêtes qui utilisent des opérateurs ou des fonctions non déterministes comme `rand()` ou `shuffle()` n'utilisent pas les résultats mis en cache. Par exemple, `LIMIT` sans `ORDER BY` est non déterministe et n'est pas mis en cache, mais `LIMIT` avec `ORDER BY` est déterministe et est mis en cache.
- La réutilisation des résultats des requêtes est prise en charge dans la console Athena, dans l'API Athena et dans le pilote JDBC. Actuellement, la prise en charge du pilote ODBC pour la réutilisation des résultats de requêtes n'est disponible que pour Windows.
- Pour utiliser la fonction de réutilisation des résultats des requêtes avec JDBC, la version minimale du pilote requise est 2.0.34.1000. Pour ODBC, la version minimale requise du pilote est 1.1.19.1002. Pour obtenir des informations sur le téléchargement du pilote, voir Connexion à Amazon Athena avec les pilotes ODBC et JDBC.
- La réutilisation des résultats de requête n'est pas prise en charge pour les requêtes qui utilisent plusieurs catalogues de données.
- La réutilisation des résultats de requête n'est pas prise en charge pour les requêtes qui incluent plus de 20 tables.

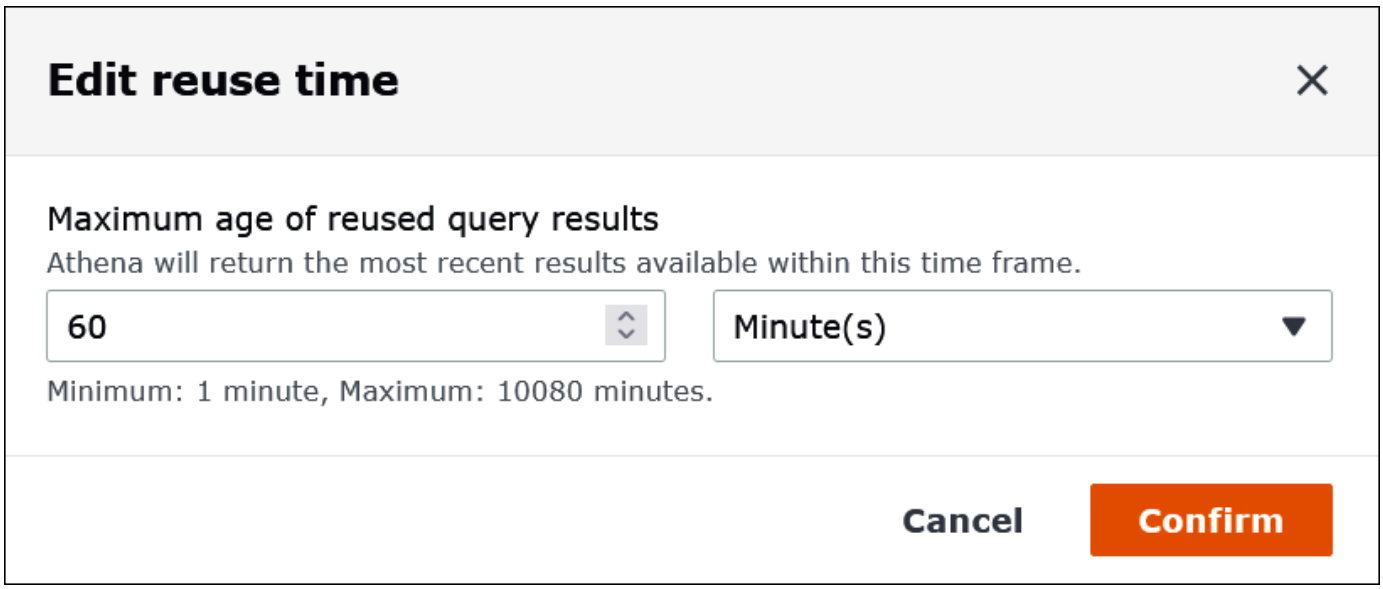
Réutilisation des résultats des requêtes dans la console Athena

Pour utiliser cette fonctionnalité, activez l'option Reuse query results (Réutiliser les résultats des requêtes) dans l'éditeur de requêtes Athena.

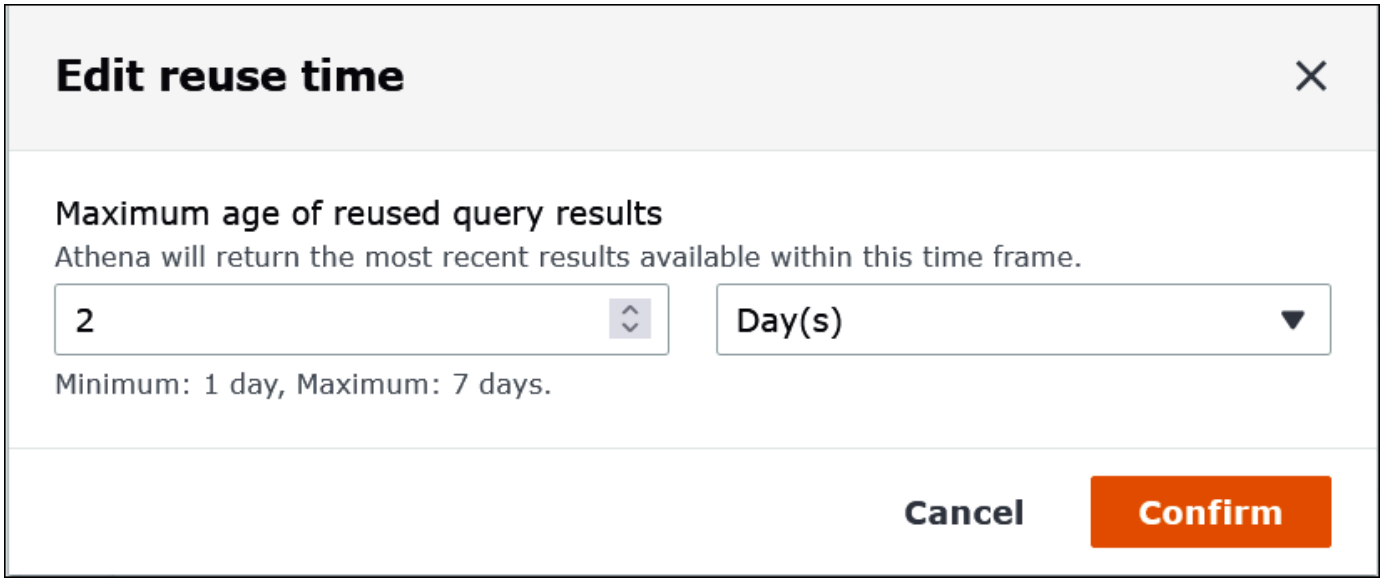


Configurer la fonctionnalité de réutilisation des résultats des requêtes

1. Dans l'éditeur de requêtes Athena, sous l'option Reuse query results (Réutiliser les résultats des requêtes), cliquez sur l'icône de modification située à côté de up to 60 minutes ago (jusqu'à 60 minutes auparavant).
2. Dans la boîte de dialogue Edit reuse time (Modifier le temps de réutilisation), dans la zone de droite, choisissez une unité de temps (minutes, heures ou jours).
3. Dans la zone de gauche, saisissez ou choisissez le nombre d'unités de temps que vous souhaitez spécifier. La durée maximale que vous pouvez saisir est l'équivalent de sept jours, quelle que soit l'unité de temps choisie.

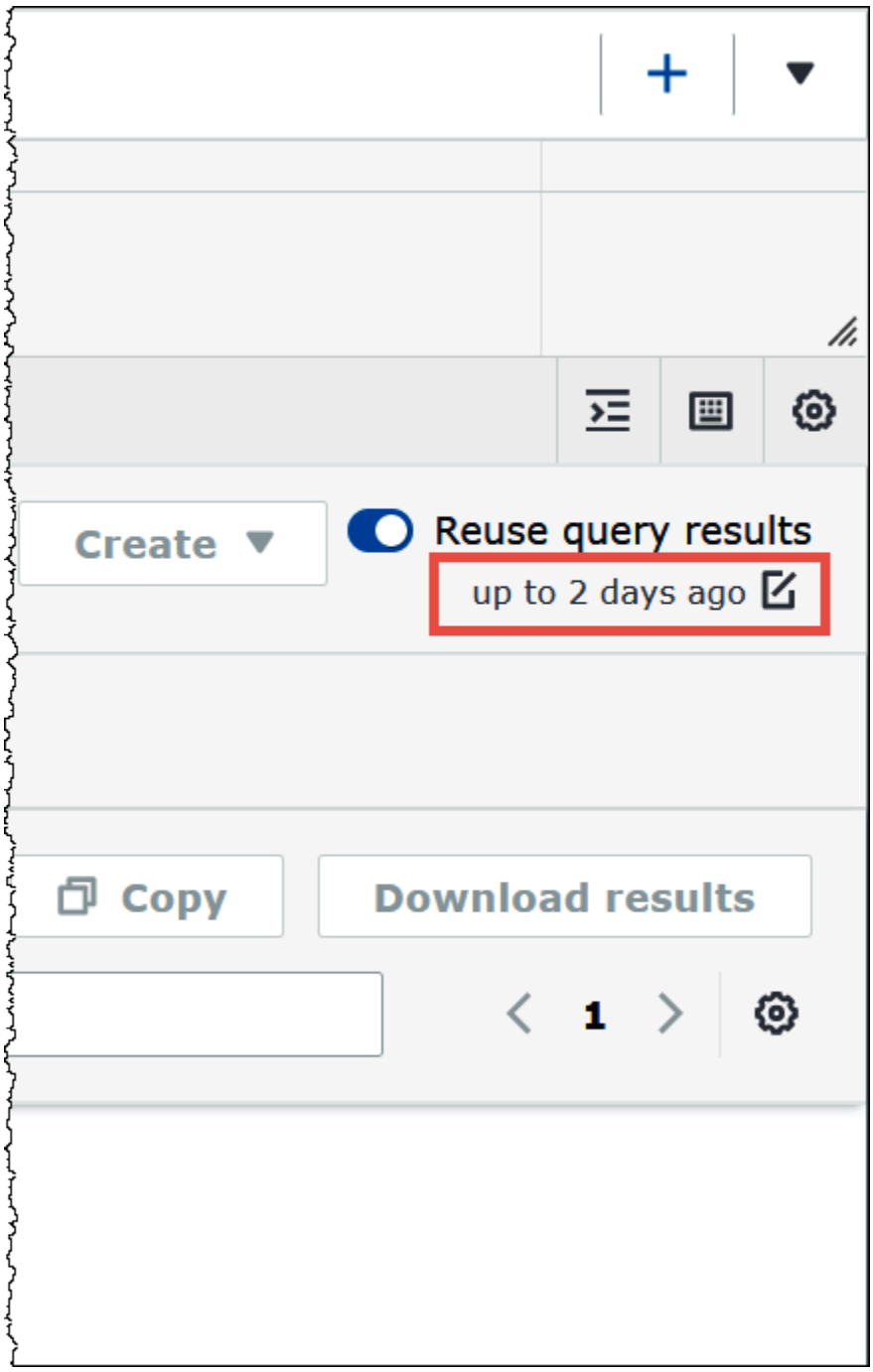


L'exemple suivant spécifie une durée de réutilisation maximale de deux jours.



4. Choisissez Confirmer.

   Une bannière confirme votre modification de configuration et l'option Reuse query results (Réutiliser les résultats des requêtes) affiche votre nouveau paramètre.

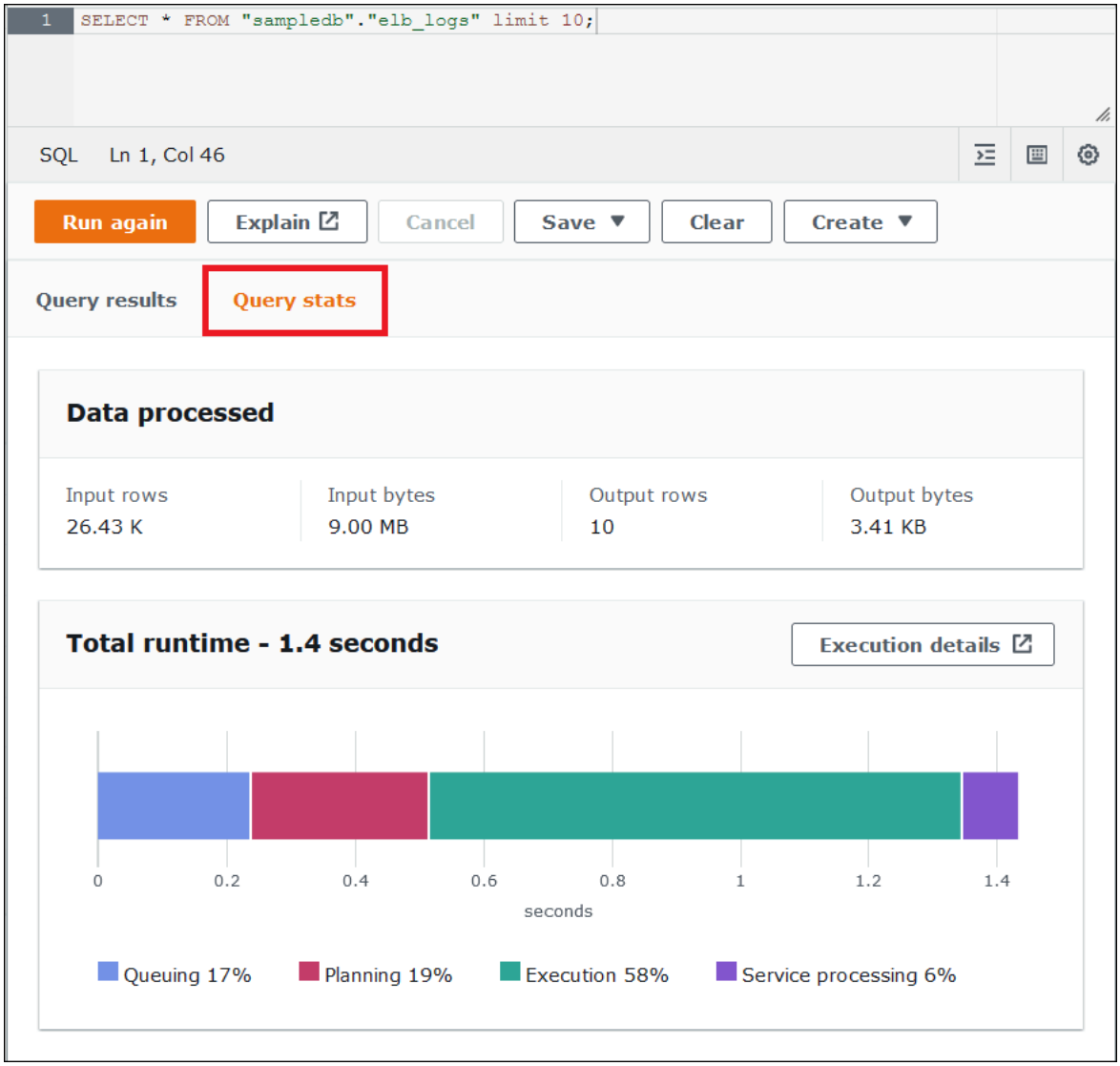


Affichage des statistiques et des détails d'exécution pour les requêtes terminées

Après avoir exécuté une requête, vous pouvez obtenir des statistiques sur les données d'entrée et de sortie traitées, voir une représentation graphique du temps nécessaire pour chaque phase de la requête et explorer de manière interactive les détails de l'exécution.

Pour afficher les statistiques d'une requête terminée

1. Après avoir exécuté une requête dans l'éditeur de requête Athena, choisissez l'onglet Query stats (Statistiques de la requête).



L'onglet Query stats (Statistiques de la requête) fournit les informations suivantes :

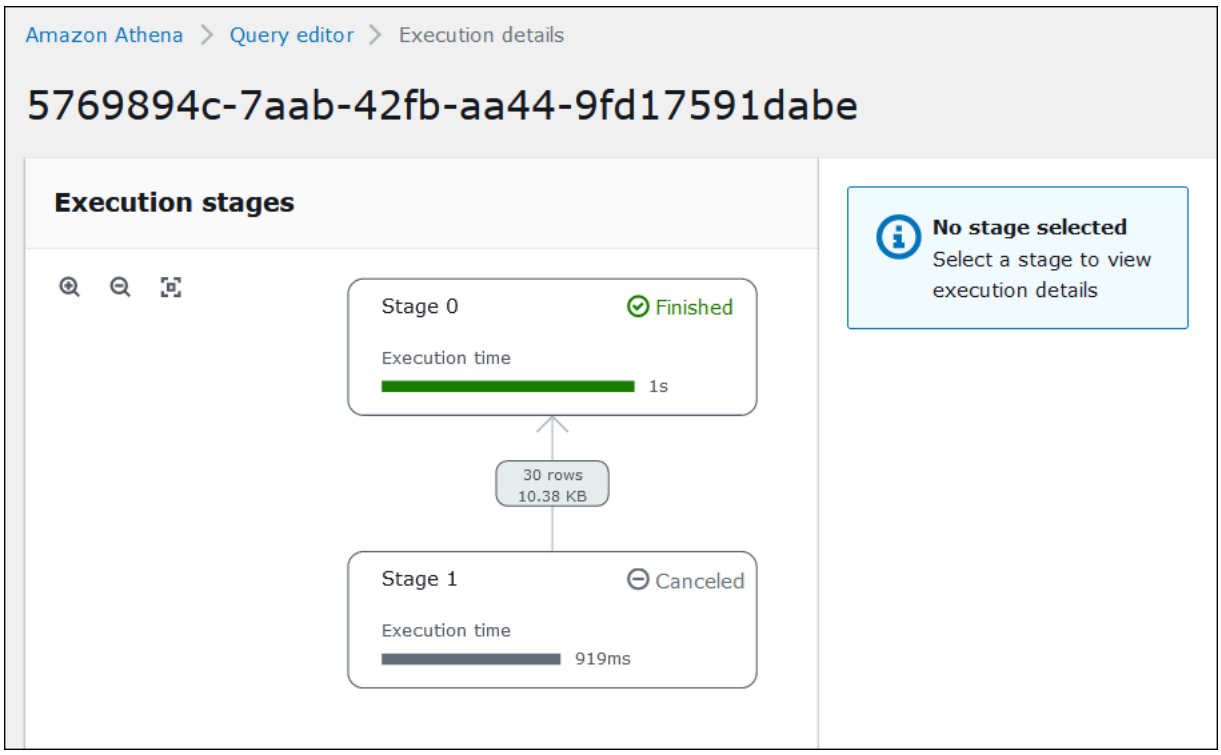
- Données traitées : affiche le nombre de lignes d'entrée et d'octets traités, ainsi que le nombre de lignes et d'octets en sortie.
- Durée totale d'exécution : affiche la durée totale d'exécution de la requête et une représentation graphique de la part de ce temps consacrée à la mise en file d'attente, à la planification, à l'exécution et au traitement des services.

> **Note**
>
> Les informations relatives au nombre de lignes d'entrée et de sortie et à la taille des données ne sont pas affichées lorsqu'une requête comporte des filtres de niveau ligne définis dans Lake Formation.

2. Pour explorer de manière interactive des informations sur la manière dont la requête a été exécutée, choisissez Execution details (Détails de l'exécution).

La page Execution details (Détails de l'exécution) affiche l'ID d'exécution de la requête et un graphique des étapes de base zéro de la requête. Les étapes sont ordonnées du début à la fin, de bas en haut. L'étiquette de chaque étape indique le temps qu'il a fallu pour que l'étape s'exécute.

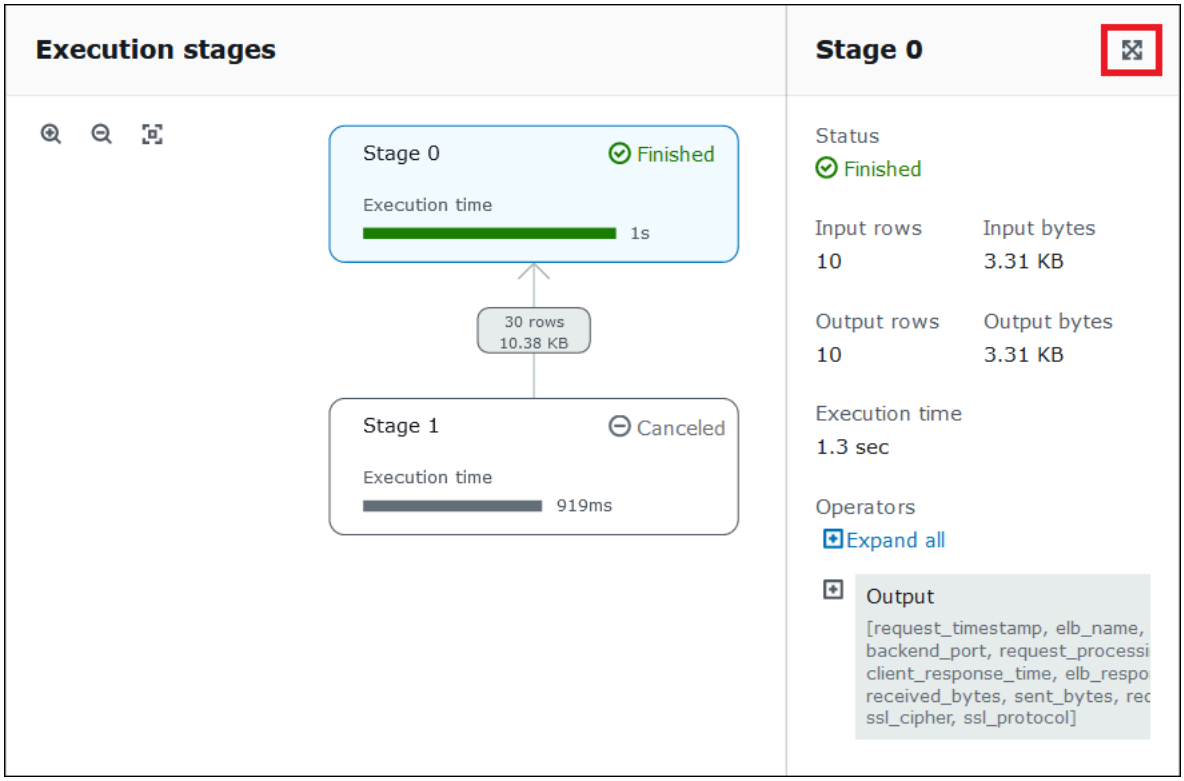
> **Note**
>
> Le temps d'exécution total et le temps d'exécution d'une requête sont souvent très différents. Par exemple, une requête dont le temps d'exécution total est exprimé en minutes peut indiquer le temps d'exécution d'une étape en heures. Comme une étape est une unité logique de calcul exécutée en parallèle sur de nombreuses tâches, le temps d'exécution d'une étape est le temps d'exécution cumulé de toutes ses tâches. Malgré cet écart, le temps d'exécution d'une étape peut être utile en tant qu'indicateur relatif pour savoir quelle étape a été la plus gourmande en calculs dans une requête.



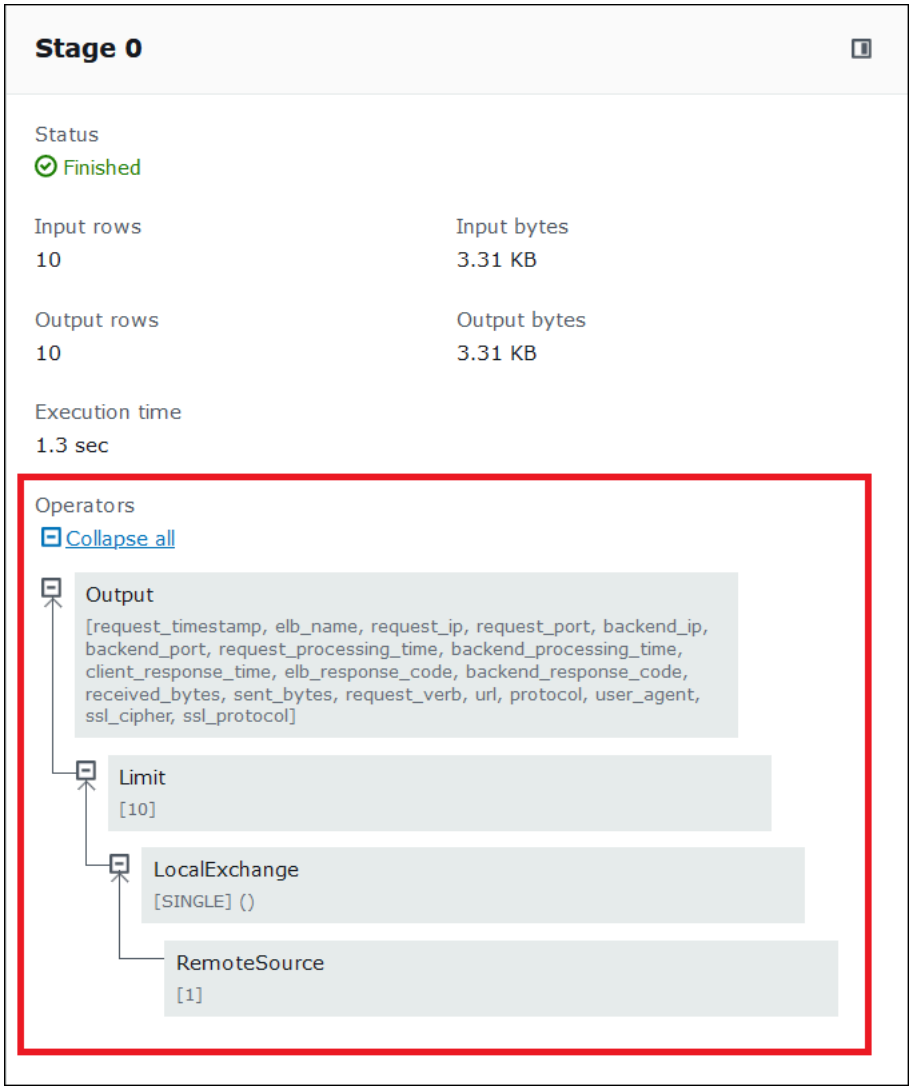
Pour parcourir le graphique, utilisez les options suivantes :

- Pour effectuer un zoom avant ou arrière, faites défiler la souris ou utilisez les icônes de grossissement.
- Pour ajuster le graphique à l'écran, choisissez l'option Zoom to fit (Zoom sur la taille).
- Pour déplacer le graphique, maintenez le pointeur de la souris et glissez-le.

3. Pour voir plus de détails sur une étape, choisissez-là. Le volet des détails de l’étape sur la droite indique le nombre de lignes et d'octets en entrée et en sortie, ainsi qu'une arborescence d'opérateurs.



4. Pour voir les détails de l'étape dans toute sa largeur, cliquez sur l'icône de développement en haut à droite du volet des détails.
5. Pour obtenir des informations sur les parties de l’étape, développez un ou plusieurs éléments dans l'arborescence des opérateurs.



Pour plus d'informations sur les détails d'exécution, consultez Explication des résultats de l'instruction EXPLAIN d'Athena.

Ressources supplémentaires

Pour plus d'informations, veuillez consulter les ressources suivantes.

Affichage des plans d'exécution pour les requêtes SQL

Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena

Utilisation des vues

Une vue dans Amazon Athena est une table logique et non physique. La requête qui définit une vue est exécutée chaque fois qu'elle est référencée dans une requête.

Vous pouvez créer une vue à partir d'une requête `SELECT`, puis référencer cette vue dans de futures requêtes. Pour plus d'informations, consultez CREATE VIEW.

Rubriques

Quand l'utiliser les vues ?

Vous avez la possibilité de créer des vues dans les cas suivants :

- Interrogation d'un sous-ensemble de données. Par exemple, vous pouvez créer une vue avec un sous-ensemble de colonnes à partir de la table d'origine pour simplifier l'interrogation des données.
- Combinaison de plusieurs tables dans une requête. Lorsque vous avez plusieurs tables et que vous souhaitez les combiner avec `UNION ALL`, vous pouvez créer une vue avec cette expression pour simplifier les requêtes sur les tables combinées.
- Masquage de la complexité des requêtes de base existantes et simplification des requêtes exécutées par les utilisateurs. Les requêtes de base incluent souvent des jointures entre les tables, les expressions dans la liste de colonnes, et d'autres syntaxes SQL, ce qui complique leur compréhension et leur débogage. Vous pouvez créer une vue qui masque la complexité et simplifie les requêtes.
- Essai avec des techniques d'optimisation et création de requêtes optimisées. Par exemple, si vous trouvez une combinaison de conditions `WHERE`, l'ordre `JOIN`, ou d'autres expressions qui illustrent les meilleures performances, vous pouvez créer une vue avec ces clauses et expressions. Les

applications peuvent ensuite créer des requêtes relativement simples pour cette vue. Si, par la suite, vous trouvez un meilleur moyen d'optimiser la requête d'origine, lorsque vous recréez la vue, toutes les applications tirent immédiatement parti de la requête de base optimisée.

- Masquage de la table et des noms de colonnes sous-jacents et minimisation des problèmes de maintenance si ces noms changent. Dans ce cas, vous recréez la vue à l'aide des nouveaux noms. Toutes les requêtes qui utilisent la vue et non les tables sous-jacentes continuent de s'exécuter sans modifications.

Actions prises en charge pour les vues dans Athena

Athena prend en charge les actions suivantes pour les vues. Vous pouvez exécuter ces commandes dans l'éditeur de requête.

Instruction	Description
CREATE VIEW	Crée une nouvelle vue à partir d'une requête `SELECT` spécifiée. Pour plus d'informations, consultez Création de vues. La clause `OR REPLACE` en option vous permet de mettre à jour la vue existante en la remplaçant.
DESCRIBE VIEW	Affiche la liste des colonnes pour la vue nommée. Vous pouvez ainsi examiner les attributs d'une vue complexe.
DROP VIEW	Supprime une vue existante. La clause `IF EXISTS` en option supprime l'erreur si la vue n'existe pas.
SHOW CREATE VIEW	Affiche l'instruction SQL qui crée la vue spécifiée.
SHOW VIEWS	Répertorie les vues dans la base de données spécifiée, ou dans la base de données actuelle, si vous ne spécifiez pas le nom de base de données. Utilisez la clause `LIKE` en option avec une expressio n régulière pour restreindre la liste des noms de vue. Vous pouvez également voir la liste des vues dans le volet gauche de la console.
SHOW COLUMNS	Affiche les colonnes du schéma d'une vue.

Considérations relatives aux vues

Les considérations suivantes s'appliquent à la création et à l'utilisation des vues dans Athena :

- Dans Athena, vous pouvez prévisualiser et utiliser les vues créées dans la console Athena, dans le AWS Glue Data Catalog, si vous avez migré pour l'utiliser, ou avec Presto exécuté sur le cluster Amazon EMR connecté au même catalogue. Vous ne pouvez pas prévisualiser ou ajouter à Athena des vues qui ont été créées autrement.
- Si vous avez créé des vues Athena dans le catalogue de données, ce dernier traite les vues comme des tables. Vous pouvez utiliser le contrôle précis de l'accès au niveau des tables dans le catalogue de données pour limiter l'accès à ces vues.
- Athena vous empêche d'exécuter des vues récursives et affiche un message d'erreur dans de tels cas. Une vue récursive est une requête de vue qui se référence elle-même.
- Athena affiche un message d'erreur lorsqu'il détecte des vues périmées. Une vue obsolète est signalée lorsque l'une des situations suivantes se produit :
  - La vue fait référence à des tables ou à des bases de données qui n'existent pas.
  - Une modification de schéma ou de métadonnées est effectuée dans une table référencée.
  - Une table référencée est supprimée et recréée avec un autre schéma ou une autre configuration.
- Vous pouvez créer et exécuter des vues imbriquées dès lors que la requête derrière la vue imbriquée est valide et que les tables et les bases de données existent.

Limitations pour les vues

- Les noms des vues Athena ne peuvent pas contenir de caractères spéciaux, autres que le trait de soulignement (`_`). Pour plus d'informations, consultez Noms des tables, des bases de données et des colonnes.
- Évitez d'utiliser des mots-clés réservés pour nommer les vues. Si vous utilisez des mots-clés réservés, utilisez des guillemets doubles pour entourer mots-clés réservés dans vos requêtes sur les vues. Consultez Mots-clés réservés.
- Vous ne pouvez pas utiliser de vues créées dans Athena avec des métastores Hive externes ou des fonctions définies par l'utilisateur. Pour plus d'informations sur l'utilisation des vues créées en externe dans Hive, consultez Utilisation des vues Hive.
- Vous ne pouvez pas utiliser de vues avec des fonctions géospatiales.
- Vous ne pouvez pas utiliser des vues pour gérer le contrôle d'accès aux données dans Simple Storage Service (Amazon S3). Pour interroger une vue, vous avez besoin d'autorisations

pour accéder aux données stockées dans Simple Storage Service (Amazon S3). Pour plus d'informations, consultez Accès à Amazon S3 depuis Athena.

- Bien que l'interrogation de vues entre comptes soit prise en charge à la fois dans les versions 2 et 3 du moteur Athena, vous ne pouvez pas créer de vue incluant un compte croisé AWS Glue Data Catalog. Pour plus d'informations sur l'accès inter-comptes aux catalogues de données, consultez Accès entre comptes aux catalogues de données AWS Glue.
- Les colonnes de métadonnées masquées Hive ou Iceberg `$bucket`, `$file_modified_time`, `$file_size` et `$partition` ne sont pas prises en charge pour les vues dans Athena. Pour plus d'informations sur l'utilisation de la colonne de métadonnées `$path` dans Athena, consultez Obtention des emplacements de fichiers pour les données source dans Simple Storage Service (Amazon S3) .

Gestion des vues dans la console

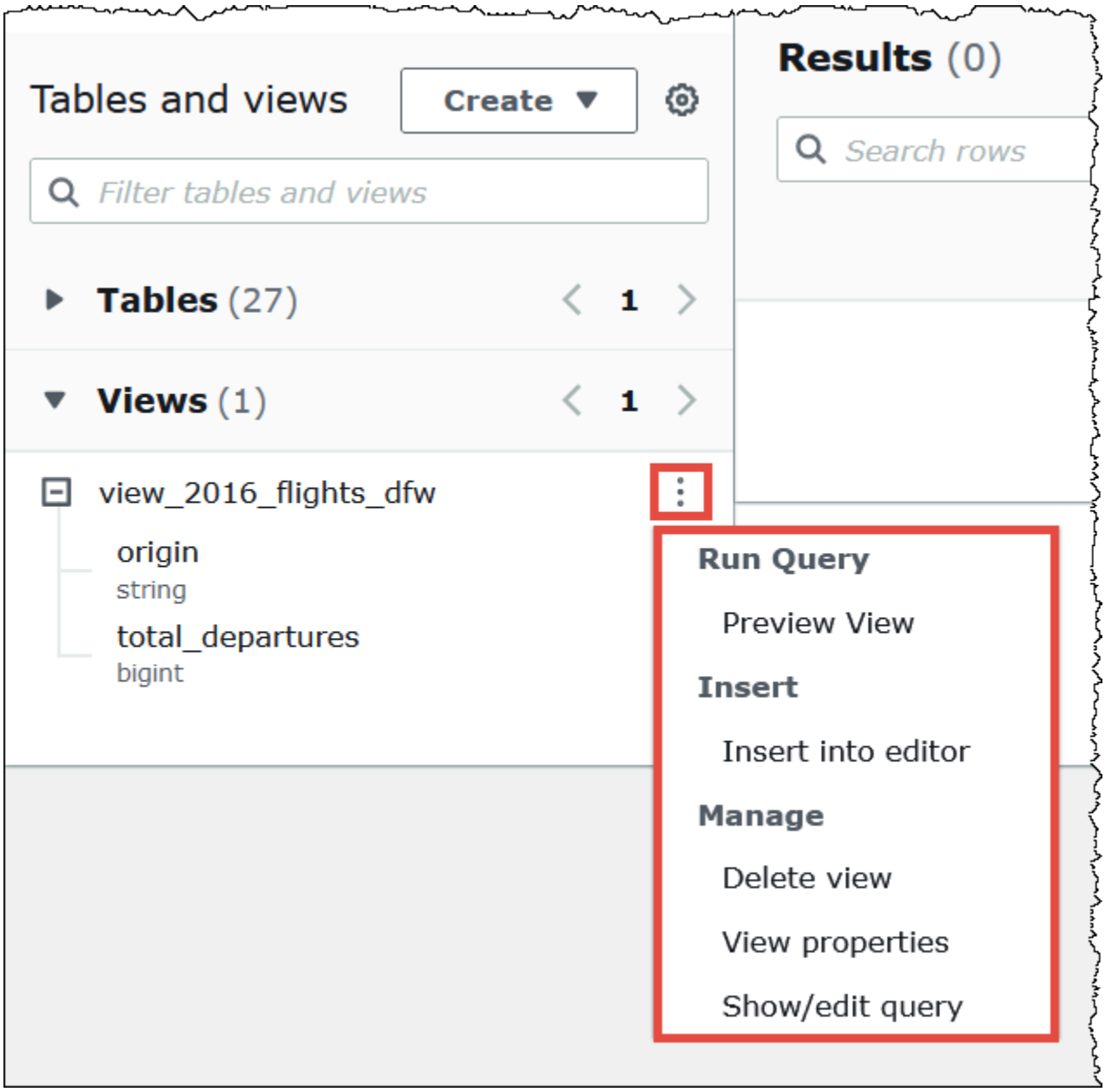
Dans la console Athena, vous pouvez :

- Localiser toutes les vues dans le volet de gauche, où les tables sont répertoriées.
- Filtrer les vues.
- Prévisualiser une vue, afficher ses propriétés, la modifier ou la supprimer.

Affichage des actions d'une vue

Une vue s'affiche dans la console uniquement si vous l'avez déjà créée.

1. Dans la console Athena, choisissez Views (Vues), puis choisissez une vue pour la développer et afficher les colonnes de la vue.
2. Choisissez les trois points verticaux à côté de la vue pour afficher la liste des actions de la vue.



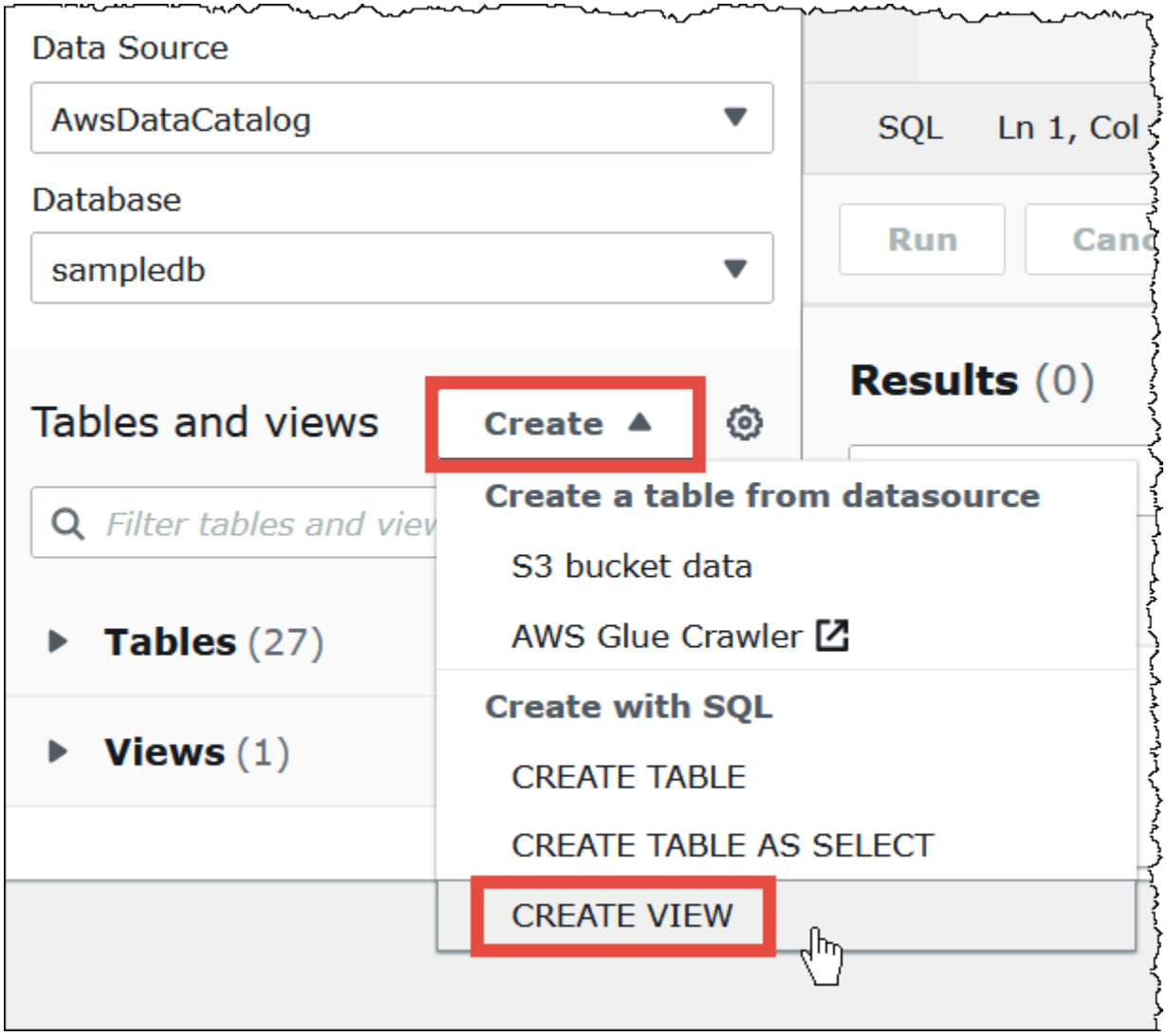
3. Choisissez des actions pour prévisualiser la vue, insérer le nom de la vue dans l'éditeur de requêtes, supprimer la vue, afficher les propriétés de la vue ou afficher et modifier la vue dans l'éditeur de requêtes.

Création de vues

Vous pouvez créer une vue dans la console Athena à l'aide d'un modèle ou en exécutant une requête existante.

Création d'une vue en utilisant un modèle

1. Dans la console Athena, près de Tables and views (Tables et vues), choisissez Create (Créer) puis Create view (Créer une vue).



   Cette action place un modèle de vue modifiable dans l'éditeur de requêtes.

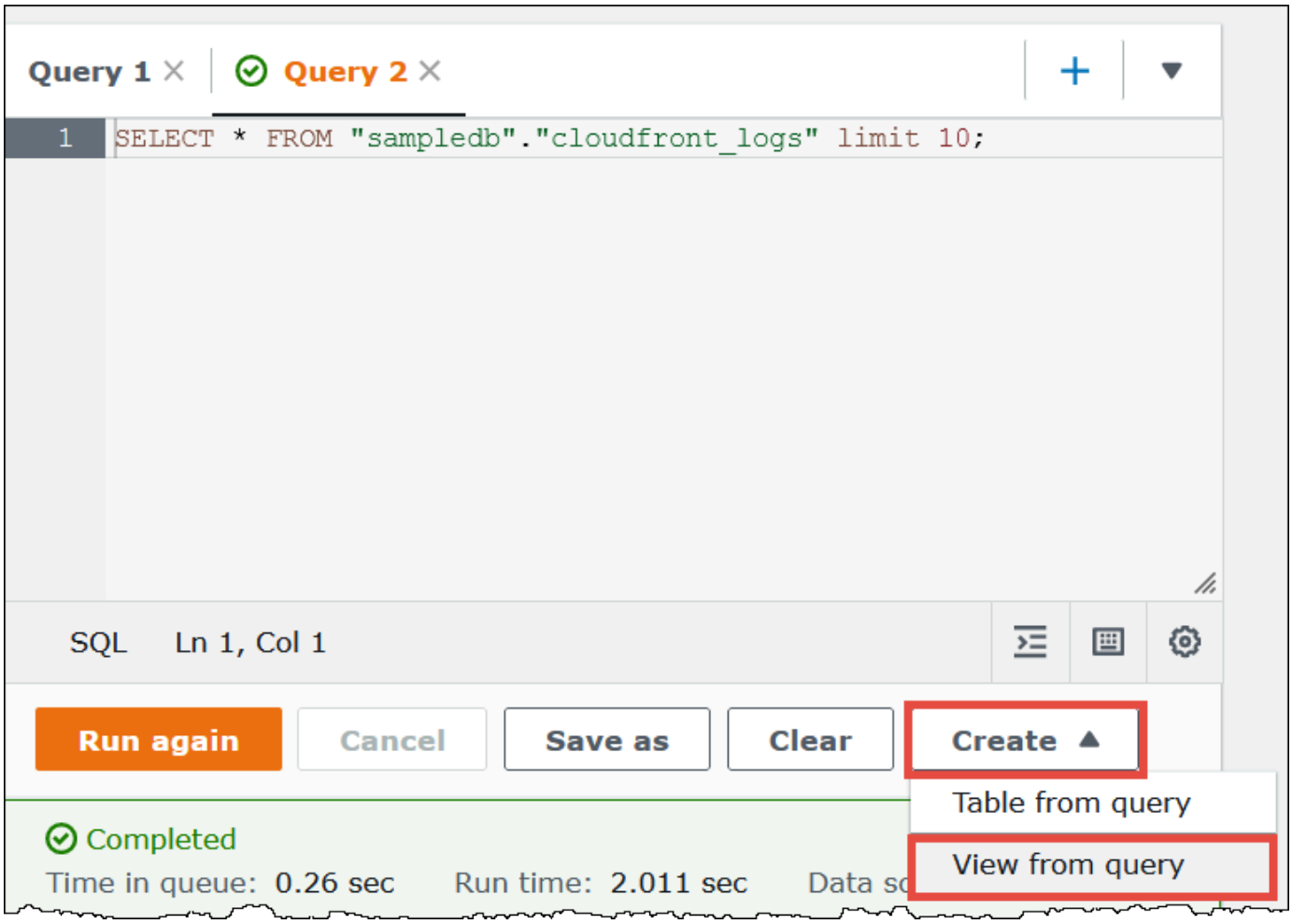
2. Modifiez le modèle de vue en fonction de vos besoins. Lorsque vous saisissez un nom pour la vue dans l'instruction, n'oubliez pas que les noms de vue ne peuvent pas contenir de caractères spéciaux autres que le trait de soulignement (_). Consultez Noms des tables, des bases de données et des colonnes. Évitez d'utiliser Mots-clés réservés pour nommer les vues.

   Pour plus d'informations sur la création de vues, consultez CREATE VIEW et Exemples de vues.

3. Choisissez Run (Exécuter) pour créer la vue. La vue apparaît dans la liste des vues de la console Athena.

Création d'une vue à partir d'une requête existante

1. Utilisez l'éditeur de requêtes Athena pour exécuter une requête existante.
2. Dans la fenêtre de l'éditeur de requêtes, choisissez Create (Créer), puis View from query (Afficher à partir d'une requête).



3. Dans la boîte de dialogue Create View (Créer une vue), saisissez un nom de vue, puis choisissez Create (Créer). Les noms de vue ne peuvent pas contenir de caractères spéciaux autres que le trait de soulignement (`_`). Consultez Noms des tables, des bases de données et des colonnes. Évitez d'utiliser Mots-clés réservés pour nommer les vues.

   Athena ajoute la vue à la liste des vues de la console et affiche l'instruction `CREATE VIEW` pour la vue dans l'éditeur de requêtes.

Remarques

- Si vous supprimez une table sur laquelle une table est basée, puis tentez d'exécuter la vue, Athena affiche un message d'erreur.
- Vous pouvez créer une vue imbriquée, qui est une vue sur une vue existante. Athena vous empêche d'exécuter une vue récursive qui fait référence à elle-même.

Exemples de vues

Pour afficher la syntaxe de la requête de vue, utilisez SHOW CREATE VIEW.

Example Exemple 1

Considérez les deux tables suivantes : une table `employees` avec deux colonnes, `id` et `name`, et une table `salaries`, avec deux colonnes, `id` et `salary`.

Dans cet exemple, nous créons une vue nommée `name_salary` en tant que requête `SELECT` qui obtient une liste d'ID mappés aux salaires à partir des tables `employees` et `salaries` :

```
CREATE VIEW name_salary AS
SELECT
 employees.name,
 salaries.salary
FROM employees, salaries
WHERE employees.id = salaries.id
```

Example Exemple 2

Dans l'exemple suivant, nous créons une vue nommée `view1` qui vous permet de masquer la syntaxe de requête plus complexe.

Cette vue s'exécute au-dessus de deux tables, `table1` et `table2`, où chaque table est une autre requête `SELECT`. La vue sélectionne les colonnes de `table1` et joint les résultats avec `table2`. La jointure est basée sur la colonne a qui est présente dans les deux tables.

```
CREATE VIEW view1 AS
WITH
  table1 AS (
        SELECT a,
```

```
        MAX(b) AS the_max
        FROM x
        GROUP BY a
        ),
  table2 AS (
        SELECT a,
        AVG(d) AS the_avg
        FROM y
        GROUP BY a)
SELECT table1.a, table1.the_max, table2.the_avg
FROM table1
JOIN table2
ON table1.a = table2.a;
```

Pour obtenir des informations sur l'interrogation des vues fédérées, consultez Interrogation des vues fédérées.

Utilisation des AWS Glue Data Catalog vues

Cette fonction est disponible en version préliminaire et susceptible d'être modifiée. Pour plus d'informations, consultez la section Bêtas et aperçus du document Conditions de service AWS.

Utilisez des AWS Glue Data Catalog vues lorsque vous souhaitez disposer d'une vue commune unique, Services AWS comme Amazon Athena et Amazon Redshift. Dans les affichages du Catalogue de données, les autorisations d'accès sont définies par l'utilisateur qui a créé l'affichage, et non par l'utilisateur qui interroge l'affichage. Cette méthode d'octroi d'autorisations est appelée sémantique du définisseur.

Les cas d'utilisation suivants montrent comment vous pouvez utiliser les affichages du Catalogue de données.

- Meilleur contrôle d'accès : vous créez un affichage qui limite l'accès aux données en fonction du niveau d'autorisation requis par l'utilisateur. Par exemple, vous pouvez utiliser des affichages du Catalogue de données pour empêcher les employés qui ne travaillent pas dans le service des ressources humaines (RH) de voir des données d'identification personnelle.
- Garantir la caractère complet des enregistrements : en appliquant certains filtres à l'affichage de votre Catalogue de données, vous vous assurez que les enregistrements de données d'un affichage du Catalogue de données sont toujours complets.

- Sécurité renforcée : dans les affichages du Catalogue de données, la définition de requête qui crée l'affichage doit être intacte pour que l'affichage soit créé. Cela rend les affichages du Catalogue de données moins vulnérables aux commandes SQL provenant d'acteurs malveillants.
- Empêcher l'accès aux tables sous-jacentes : la sémantique du définisseur permet aux utilisateurs d'accéder à un affichage sans mettre la table sous-jacente à leur disposition. Seul l'utilisateur qui définit l'affichage nécessite d'accéder aux tables.

Les définitions des affichages du Catalogue de données sont stockées dans l' AWS Glue Data Catalog. Cela signifie que vous pouvez utiliser AWS Lake Formation pour accorder l'accès via des autorisations de ressources, des autorisations de colonnes ou des contrôles d'accès basés sur des balises. Pour plus d'informations sur l'octroi et la révocation de l'accès à Lake Formation, consultez la rubrique Granting and revoking permissions on Data Catalog resources dans le Guide du développeur AWS Lake Formation .

Autorisations

Les affichages du Catalogue de données nécessitent trois rôles : `Lake Formation Admin`, `Definer` et `Invoker`.

- **`Lake Formation Admin`** : peut configurer toutes les autorisations de Lake Formation.
- **`Definer`** : crée l'affichage du Catalogue de données. Le rôle `Definer` doit disposer d'autorisations `SELECT` octroyables complètes pouvant être accordées sur toutes les tables sous-jacentes auxquelles la définition de l'affichage fait référence.
- **`Invoker`** : peut interroger l'affichage du Catalogue de données ou vérifier ses métadonnées. Pour afficher l'invocateur d'une requête, vous pouvez utiliser la fonction `invoker_principal()` DML. Pour plus d'informations, consultez invoqueur_principal ().

Les relations de confiance du `Definer` rôle doivent permettre l'`sts:AssumeRole`action pour les responsables du service Lake Formation AWS Glue et ceux de Lake Formation, comme dans l'exemple suivant.

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

```
            "Principal": {
                "Service": [
                    "glue.amazonaws.com",
                    "lakeformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Les autorisations IAM pour accéder à Athena sont également requises. Pour de plus amples informations, consultez AWS politiques gérées pour Amazon Athena.

Limites

- Les affichages du Catalogue de données ne peuvent pas référencer d'autres affichages.
- Vous pouvez référencer jusqu'à 10 tables dans la définition de l'affichage.
- Les tables sous-jacentes doivent être enregistrées dans Lake Formation.
- Le principal `DEFINER` ne peut être qu'un rôle IAM.
- Le rôle `DEFINER` doit disposer d'autorisations (octroyables) `SELECT` complètes sur les tables sous-jacentes.
- Les affichages `UNPROTECTED` du Catalogue de données ne sont pas pris en charge.
- Les fonctions définies par l'utilisateur (UDF) ne sont pas prises en charge dans la définition d'affichage.
- Les sources de données fédérées Athena ne peuvent pas être utilisées dans les affichages du Catalogue de données.
- Les affichages du Catalogue de données ne sont pas pris en charge pour les métastores Hive externes.
- Athena affiche un message d'erreur lorsqu'il détecte des vues périmées. Une vue obsolète est signalée lorsque l'une des situations suivantes se produit :
  - La vue fait référence à des tables ou à des bases de données qui n'existent pas.
  - Une modification de schéma ou de métadonnées est effectuée dans une table référencée.
  - Une table référencée est supprimée et recréée avec un autre schéma ou une autre configuration.

Création d'un affichage du Catalogue de données

L'exemple de syntaxe suivant montre comment un utilisateur du rôle `Definer` crée l'affichage `orders_by_date` du Catalogue de données. L'exemple suppose que le rôle `Definer` dispose d'autorisations `SELECT` complètes sur la table `orders` de la base de données `default`.

```
CREATE PROTECTED MULTI DIALECT VIEW orders_by_date
SECURITY DEFINER
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
WHERE order_city = 'SEATTLE'
GROUP BY orderdate
```

Interrogation d'un affichage du Catalogue de données

Une fois l'affichage créé, le `Lake Formation Admin` peut accorder des autorisations `SELECT` sur l'affichage du Catalogue de données aux principaux `Invoker`. Les principaux `Invoker` peuvent ensuite interroger l'affichage sans avoir accès aux tables de base sous-jacentes référencées par l'affichage. Voici un exemple de requête `Invoker`.

```
SELECT * from orders_by_date where price > 5000
```

Mise à jour d'un affichage du Catalogue de données

Le `Lake Formation Admin` ou le `Definer` peuvent utiliser la syntaxe `ALTER VIEW UPDATE DIALECT` pour mettre à jour la définition de l'affichage. L'exemple suivant modifie la définition de l'affichage pour sélectionner des colonnes dans la table `returns` plutôt que dans la table `orders`.

```
ALTER VIEW orders_by_date UPDATE DIALECT
AS
SELECT return_date, sum(totalprice) AS price
FROM returns
WHERE order_city = 'SEATTLE'
GROUP BY orderdate
```

Pour plus d'informations sur la syntaxe de création et de gestion des affichages du Catalogue de données, consultez Syntaxe d'affichage du Catalogue de données Glue.

Syntaxe d’affichage du Catalogue de données Glue

> Cette fonction est disponible en version préliminaire et susceptible d'être modifiée. Pour plus d'informations, consultez la section Bêtas et aperçus du document Conditions de service AWS.

Cette section décrit les commandes DDL (Data Definition Language) permettant de créer et de gérer des AWS Glue Data Catalog vues.

ALTER VIEW DIALECT

Vous pouvez mettre à jour les affichages du Catalogue de données soit en ajoutant un dialecte de moteur, soit en mettant à jour ou en supprimant un dialecte de moteur existant. Seuls le `LakeFormationAdmin` et le `Definer` (l’utilisateur qui a créé l’affichage) sont autorisés à utiliser l’instruction `ALTER VIEW DIALECT` dans un affichage du Catalogue de données.

Syntaxe

```
ALTER VIEW name [ FORCE ] [ ADD|UPDATE ] DIALECT AS query
```

```
ALTER VIEW name [ DROP ] DIALECT
```

FORCE

Le mot-clé `FORCE` entraîne le remplacement des informations du dialecte de moteur contradictoires dans un affichage par la nouvelle définition. Le mot-clé `FORCE` est utile lorsqu’une mise à jour d’un affichage du Catalogue de données entraîne des définitions d’affichage contradictoires entre les dialectes de moteur existants. Supposons qu’un affichage du Catalogue de données utilise à la fois les dialectes Athena et Amazon Redshift et que la mise à jour entraîne un conflit avec Amazon Redshift dans la définition de l’affichage. Dans ce cas, vous pouvez utiliser le mot-clé `FORCE` pour autoriser la mise à jour et marquer le dialecte Amazon Redshift comme obsolète. Lorsque les moteurs marqués comme obsolètes interrogent l’affichage, la requête échoue. Les moteurs déclenchent une exception pour interdire les résultats obsolètes. Pour corriger cela, mettez à jour les dialectes obsolètes dans l’affichage.

ADD

Ajoute un nouveau dialecte de moteur à l’affichage du Catalogue de données. Le moteur spécifié ne doit pas déjà exister dans l’affichage du Catalogue de données.

UPDATE

Met à jour un dialecte de moteur qui existe déjà dans l'affichage du Catalogue de données.

DROP

Supprime un dialecte de moteur existant d'un affichage du Catalogue de données. Une fois que vous avez supprimé un moteur d'un affichage du Catalogue de données, l'affichage du Catalogue de données ne peut pas être interrogé par le moteur qui a été supprimé. Les autres dialectes de moteur présents dans l'affichage peuvent toujours interroger l'affichage.

DIALECT AS

Introduit une requête SQL spécifique au moteur.

Exemples

```
ALTER VIEW orders_by_date FORCE ADD DIALECT
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate
```

```
ALTER VIEW orders_by_date FORCE UPDATE DIALECT
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate
```

```
ALTER VIEW orders_by_date DROP DIALECT
```

CREATE PROTECTED MULTI DIALECT VIEW

Crée une vue du catalogue de données dans le AWS Glue Data Catalog. Un affichage du Catalogue de données est un schéma d'affichage unique qui fonctionne parfaitement avec Athena et d'autres moteurs SQL tels qu'Amazon Redshift et Amazon EMR.

Syntaxe

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
 [ SECURITY DEFINER ]
```

```
AS query
```

PROTECTED

Mot-clé requis. Spécifie que l'affichage est protégé contre les fuites de données. Les affichages du Catalogue de données ne peuvent être créés qu'en tant qu'affichages `PROTECTED`.

MULTI DIALECT

Spécifie que l'affichage prend en charge les dialectes SQL des différents moteurs de requête et peut donc être lu par ces moteurs.

SECURITY DEFINER

Spécifie que la sémantique du définisseur est en vigueur pour cet affichage. La sémantique du définisseur signifie que les autorisations de lecture effectives sur les tables sous-jacentes appartiennent au principal ou au rôle qui a défini l'affichage, et non au principal qui effectue la lecture proprement dite.

OR REPLACE

Un affichage du Catalogue de données ne peut pas être remplacé si des dialectes SQL provenant d'autres moteurs sont présents dans l'affichage. Si le moteur appelant possède le seul dialecte SQL présent dans l'affichage, celui-ci peut être remplacé.

Exemple

L'exemple suivant crée l'affichage `orders_by_date` du Catalogue de données basé sur une requête sur la table `orders`.

```
CREATE PROTECTED MULTI DIALECT VIEW orders_by_date
SECURITY DEFINER
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
WHERE order_city = 'SEATTLE'
GROUP BY orderdate
```

DESCRIBE

Affiche la liste des colonnes pour l'affichage du Catalogue de données spécifié. L'instruction `DESCRIBE` est similaire à l'instruction `DESCRIBE` pour les affichages Athena. Contrairement aux

affichages Athena, le résultat de l'instruction est contrôlé par le biais du contrôle d'accès à Lake Formation. Le résultat de cette requête ne correspond donc pas à toutes les colonnes de l'affichage, mais uniquement aux colonnes auxquelles l'appelant a accès.

Syntaxe

```
DESCRIBE [db_name.]view_name
```

Exemples

```
DESCRIBE orders
```

DROP VIEW

Supprime un affichage du Catalogue de données uniquement si le dialecte du moteur appelant est présent dans l'affichage du Catalogue de données. Par exemple, si un utilisateur appelle `DROP VIEW` depuis Athena, l'affichage est supprimé uniquement si le dialecte d'Athena existe dans l'affichage. Sinon, l'opération échoue. Seuls le Lake Formation Admin et le définisseur de l'affichage sont autorisés à utiliser l'instruction `DROP VIEW` dans un affichage du Catalogue de données.

Syntaxe

```
DROP VIEW [ IF EXISTS ] view_name
```

Exemples

```
DROP VIEW orders_by_date
```

```
DROP FORCE VIEW IF EXISTS orders_by_date
```

La clause `IF EXISTS` en option génère l'erreur à supprimer si la vue n'existe pas.

SHOW COLUMNS

Affiche uniquement les noms de colonnes d'un seul affichage du Catalogue de données spécifié. L'instruction `SHOW COLUMNS` est similaire à l'instruction `SHOW COLUMNS` pour les affichages Athena. Contrairement aux affichages Athena, le résultat de l'instruction est contrôlé par le biais du contrôle d'accès à Lake Formation. Le résultat de cette requête ne correspond donc pas à toutes les colonnes de l'affichage, mais uniquement aux colonnes auxquelles l'appelant a accès.

Syntaxe

```
SHOW COLUMNS {FROM|IN} database_name.view_name
```

```
SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name]
```

SHOW CREATE VIEW

Indique la syntaxe SQL qui a créé l'affichage du Catalogue de données. Le code SQL renvoyé indique la syntaxe de création d'affichage utilisée dans Athena. Seuls les principaux Lake Formation Admin et définisseur de l'affichage sont autorisés à appeler `SHOW CREATE VIEW` sur un affichage du Catalogue de données.

Syntaxe

```
SHOW CREATE VIEW view_name
```

Exemples

```
SHOW CREATE VIEW orders_by_date
```

SHOW VIEWS

Répertorie les noms de tous les affichages de la base de données. Tous les affichages du Catalogue de données de la base de données qui utilisent le dialecte SQL du moteur Athena sont répertoriés. Les autres affichages du Catalogue de données dans lesquels le dialecte du moteur Athena n'est pas présent sont filtrés.

Syntaxe

```
SHOW VIEWS [IN database_name] [LIKE 'regular_expression']
```

Exemples

```
SHOW VIEWS
```

```
SHOW VIEWS IN marketing_analytics LIKE 'orders*'
```

Utilisation de requêtes enregistrées

Vous pouvez utiliser la console Athena pour enregistrer, modifier, exécuter, renommer et supprimer les requêtes que vous créez dans l'éditeur de requêtes.

Considérations et restrictions

- Vous pouvez mettre à jour le nom, la description et le texte de requête des requêtes enregistrées.
- Vous pouvez uniquement mettre à jour les requêtes dans votre propre compte.
- Vous ne pouvez pas modifier le groupe de travail ou la base de données auquel la requête appartient.
- Athena ne conserve pas d'historique des modifications de requêtes. Si vous souhaitez conserver une version particulière d'une requête, enregistrez-la sous un autre nom.

Utilisation de requêtes enregistrées dans la console Athena

Pour enregistrer une requête et lui attribuer un nom

1. Dans l'éditeur de requêtes de la console Athena, saisissez ou exécutez une requête.
2. Au-dessus de la fenêtre de l'éditeur de requêtes, sur l'onglet de la requête, choisissez les trois points verticaux, puis choisissez Save as (Enregistrer sous).
3. Dans la boîte de dialogue Choose a name (Choisir un nom), saisissez un nom pour la requête et une description facultative. Vous pouvez utiliser la fenêtre extensible Preview SQL query (Aperçu de la requête SQL) pour vérifier le contenu de la requête avant de l'enregistrer.
4. Choisissez Save query (Enregistrer la requête).

   Dans l'éditeur de requêtes, l'onglet de la requête affiche le nom que vous avez spécifié.

Pour exécuter une requête enregistrée

1. Dans la console Athena, choisissez l'onglet Saved queries (Requêtes enregistrées).
2. Dans la liste Saved queries (Requêtes enregistrées), choisissez l'ID de la requête que vous souhaitez exécuter.

   L'éditeur de requêtes affiche la requête que vous avez choisie.
3. Cliquez sur Run (Exécuter).

Pour modifier une requête enregistrée

1. Dans la console Athena, choisissez l'onglet Saved queries (Requêtes enregistrées).
2. Dans la liste Saved queries (Requêtes enregistrées), choisissez l'ID de la requête que vous souhaitez modifier.
3. Modifiez la requête dans l'éditeur de requêtes.
4. Effectuez l'une des étapes suivantes :

   - Pour exécuter la requête, choisissez ensuite Run (Exécuter).
   - Pour enregistrer la requête, cliquez sur les trois points verticaux de l'onglet de la requête, puis choisissez Save as (Enregistrer sous).
   - Pour enregistrer la requête sous un autre nom, cliquez sur les trois points verticaux de l'onglet de la requête, puis choisissez Save as (Enregistrer sous).

Renommer ou supprimer une requête enregistrée déjà affichée dans l'éditeur de requêtes

1. Choisissez les trois points verticaux de l'onglet de la requête, puis choisissez Rename (Renommer) ou Delete (Supprimer).
2. Suivez les instructions pour renommer ou supprimer la requête.

Renommer une requête enregistrée non affichée dans l'éditeur de requêtes

1. Dans la console Athena, choisissez l'onglet Saved queries (Requêtes enregistrées).
2. Cochez la case en regard de la requête que vous voulez renommer.
3. Choisissez Rename (Renommer).
4. Dans Rename query (Renommer la requête), modifiez le nom et la description de la requête. Vous pouvez utiliser la fenêtre extensible Preview SQL query (Aperçu de la requête SQL)pour vérifier le contenu de la requête avant de la renommer.
5. Choisissez Rename query (Renommer la requête).

   La requête renommée apparaît dans la liste Saved queries (Requêtes enregistrées).

Supprimer une requête enregistrée non affichée dans l'éditeur de requêtes

1. Dans la console Athena, choisissez l'onglet Saved queries (Requêtes enregistrées).

2. Cochez une ou plusieurs cases pour les requêtes que vous souhaitez supprimer.
3. Choisissez Supprimer.
4. À l'invite de confirmation, choisissez Delete (Supprimer).

   Une ou plusieurs requêtes sont supprimées de la liste Saved queries (Requêtes enregistrées).

Utilisation de l'API Athena pour mettre à jour les requêtes enregistrées

Pour plus d'informations sur l'utilisation de l'API Athena pour mettre à jour une requête enregistrée, veuillez consulter l'action UpdateNamedQuery dans la référence de l'API Athena.

Utilisation de requêtes paramétrées

Vous pouvez utiliser des requêtes paramétrées Athena pour réexécuter la même requête avec des valeurs de paramètres différentes au moment de l'exécution et aider à prévenir les attaques par injection SQL. Dans Athena, les requêtes paramétrées peuvent prendre la forme de paramètres d'exécution dans n'importe quelle requête DML ou instructions préparées SQL.

- Les requêtes avec des paramètres d'exécution peuvent être effectuées en une seule étape et ne sont pas spécifiques à un groupe de travail. Vous placez des points d'interrogation dans n'importe quelle requête DML pour les valeurs que vous souhaitez paramétrer. Lorsque vous exécutez la requête, vous déclarez les valeurs des paramètres d'exécution de manière séquentielle. La déclaration des paramètres et l'attribution de valeurs pour les paramètres peuvent être effectuées dans la même requête, mais de manière découplée. Contrairement aux instructions préparées, vous pouvez sélectionner le groupe de travail lorsque vous soumettez une requête avec des paramètres d'exécution.
- Les instructions préparées nécessitent deux instructions SQL distinctes :`PREPARE` et `EXECUTE`. Tout d'abord, vous définissez les paramètres dans l'instruction `PREPARE`. Ensuite, vous lancez une instruction `EXECUTE` qui fournit les valeurs des paramètres que vous avez définis. Les instructions préparées sont spécifiques au groupe de travail ; vous ne pouvez pas les exécuter en dehors du contexte du groupe de travail auquel elles appartiennent.

Considérations et restrictions

- Les requêtes paramétrées sont prises en charge dans la version 2 du moteur Athena et les versions ultérieures. Pour plus d'informations sur les versions du moteur Athena, voir Gestion des versions du moteur Athena.

- Actuellement, les requêtes paramétrées ne sont prises en charge que pour les instructions `SELECT`, `INSERT INTO`, `CTAS` et `UNLOAD`.
- Dans les requêtes paramétrées, les paramètres sont positionnels et sont désignés par `?`. Les paramètres se voient attribuer des valeurs en fonction de leur ordre dans la requête. Les paramètres nommés ne sont pas pris en charge.
- Actuellement, les paramètres `?` ne peuvent être placés que dans la clause `WHERE`. Les syntaxes comme `SELECT ? FROM table` ne sont pas prises en charge.
- Les paramètres de point d'interrogation ne peuvent pas être placés entre guillemets doubles ou simples (c'est-à-dire, `'?'` et `"?"` ne sont pas des syntaxes valides).
- Pour que les paramètres d'exécution SQL soient traités comme des chaînes, ils doivent être placés entre guillemets simples plutôt que entre guillemets doubles.
- Si nécessaire, vous pouvez utiliser la fonction `CAST` lorsque vous entrez la valeur d'un terme paramétré. Par exemple, si vous avez une colonne du type `date` que vous avez paramétré dans une requête et que vous souhaitez rechercher la date `2014-07-05`, la saisie de `CAST('2014-07-05' AS DATE)` pour la valeur du paramètre renverra le résultat.
- Les instructions préparées sont spécifiques au groupe de travail, et les noms des instructions préparées doivent être uniques au sein du groupe de travail.
- Des autorisations IAM pour les instructions préparées sont requises. Pour plus d'informations, consultez Autorisation d'accès aux instructions préparées.
- Les requêtes avec des paramètres d'exécution dans la console Athena sont limitées à un maximum de 25 points d'interrogation.

Interrogation utilisant les paramètres d'exécution

Vous pouvez utiliser des espaces réservés de point d'interrogation dans n'importe quelle requête DML pour créer une requête paramétrée sans créer d'instruction préparée au préalable. Pour exécuter ces requêtes, vous pouvez utiliser la console Athena AWS CLI ou le SDK AWS et déclarer les variables dans l'argument. `execution-parameters`

Exécution de requêtes avec paramètres d'exécution dans la console Athena

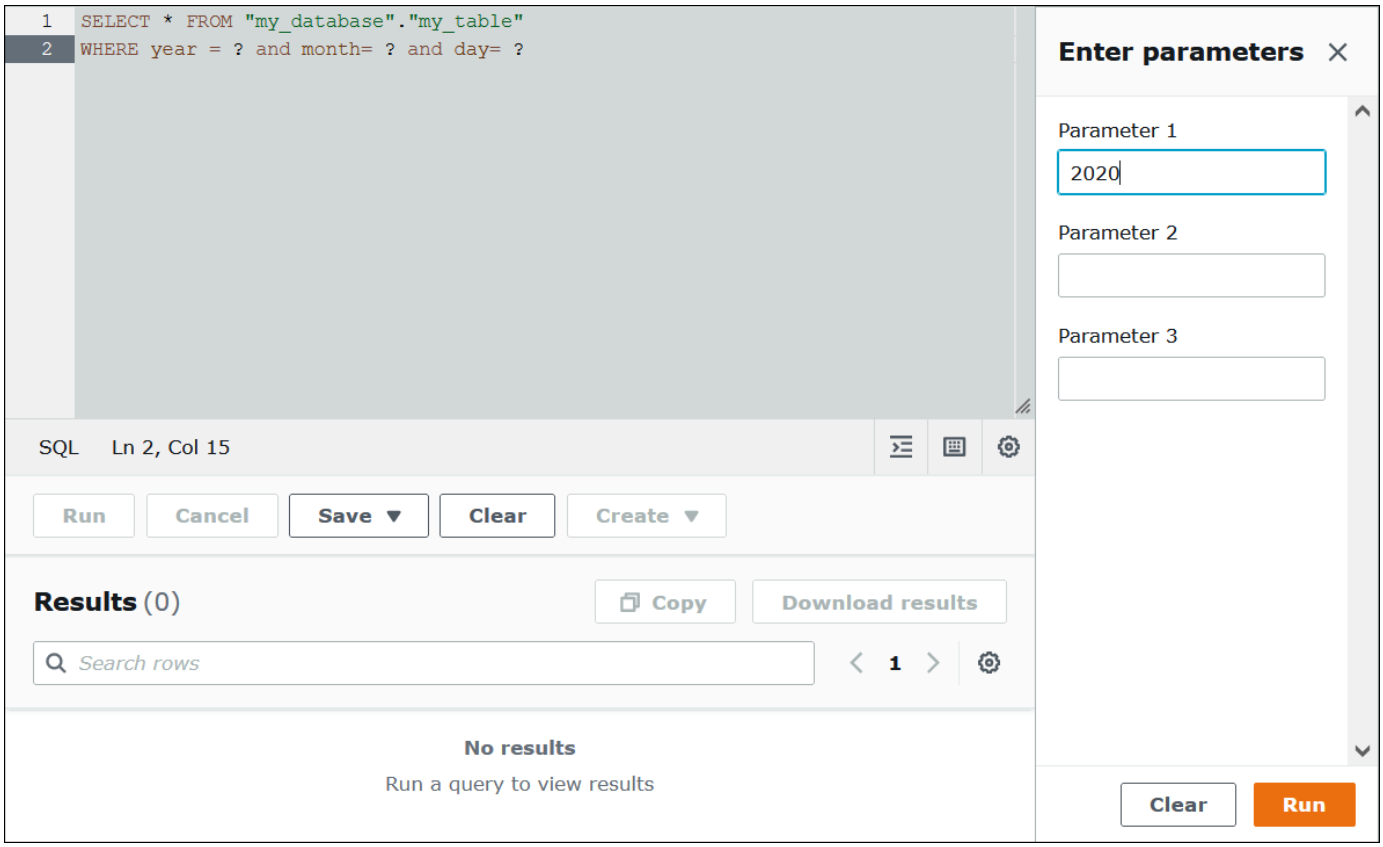
Lorsque vous exécutez une requête paramétrée qui possède des paramètres d'exécution (points d'interrogation) dans la console Athena, vous êtes invité à saisir les valeurs dans l'ordre dans lequel les points d'interrogation apparaissent dans la requête.

Pour exécuter une requête contenant des paramètres d'exécution

1. Saisissez une requête avec des espaces réservés de point d'interrogation dans l'éditeur Athena, comme dans l'exemple suivant.

```
SELECT * FROM "my_database"."my_table"
WHERE year = ? and month= ? and day= ?
```

2. Cliquez sur Exécuter.
3. Dans Enter parameters (Saisir des paramètres), saisissez une valeur dans l'ordre pour chacun des points d'interrogation de la requête.



4. Lorsque vous avez fini de saisir les paramètres, choisissez Run (Exécuter). L'éditeur affiche les résultats de la requête pour les valeurs de paramètres que vous avez saisies.

À ce stade, vous pouvez réaliser l'une des actions suivantes :

- Saisissez différentes valeurs de paramètres pour la même requête, puis choisissez Run again (Exécutez à nouveau).
- Pour effacer toutes les valeurs que vous avez saisies en même temps, choisissez Clear (Effacer).

- Pour modifier directement la requête (par exemple, pour ajouter ou supprimer des points d'interrogation), fermez d'abord la boîte de dialogue Enter parameters (Saisir des paramètres).
- Pour enregistrer la requête paramétrée pour une utilisation ultérieure, choisissez Save (Enregistrer) ou Save as (Enregistrer sous), puis donnez un nom à la requête. Pour plus d'informations sur l'utilisation de requêtes enregistrées, consultez Utilisation de requêtes enregistrées.

Pour plus de commodité, la boîte de dialogue Enter parameter (Saisir des paramètres) mémorise les valeurs que vous avez saisies précédemment pour la requête tant que vous utilisez le même onglet dans l'éditeur de requête.

Exécution de requêtes avec des paramètres d'exécution à l'aide du AWS CLI

AWS CLI Pour exécuter des requêtes avec des paramètres d'exécution, utilisez la `start-query-execution` commande et fournissez une requête paramétrée dans l'`query-string`argument. Ensuite, dans l'argument `execution-parameters`, fournissez les valeurs des paramètres d'exécution. L'exemple suivant illustre cette technique.

```
aws athena start-query-execution
--query-string "SELECT * FROM table WHERE x = ? AND y = ?"
--query-execution-context "Database"="default"
--result-configuration "OutputLocation"="s3://DOC-EXAMPLE-BUCKET;/..."
--execution-parameters "1" "2"
```

Interrogation avec des instructions préparées

Vous pouvez utiliser une instruction préparée pour l'exécution répétée d'une même requête avec des paramètres de requête différents. Une instruction préparée contient des paramètres dont les valeurs sont fournies au moment de l'exécution.

> **Note**
>
> Le nombre maximal d'instructions préparées dans un groupe de travail est de 1 000.

Instructions SQL

Vous pouvez utiliser les instructions SQL `PREPARE`, `EXECUTE` et `DEALLOCATE PREPARE` pour exécuter des requêtes paramétrées dans l'éditeur de requête de la console Athena.

- Pour spécifier des paramètres là où vous utiliseriez normalement des valeurs littérales, utilisez des points d'interrogation dans l'instruction `PREPARE`.
- Pour remplacer les paramètres par des valeurs lorsque vous exécutez la requête, utilisez la clause `USING` dans l'instruction `EXECUTE`.
- Pour supprimer une instruction préparée des instructions préparées dans un groupe de travail, utilisez l'instruction `DEALLOCATE PREPARE`.

Les sections suivantes fournissent des détails supplémentaires sur chacune de ces instructions.

PREPARE

Prépare une instruction à exécuter ultérieurement. Les instructions préparées sont enregistrées dans le groupe de travail actif avec le nom que vous spécifiez. L'instruction peut inclure des paramètres à la place des libellés qui seront remplacés lors de l'exécution de la requête. Les paramètres à remplacer par des valeurs sont signalés par des points d'interrogation.

Syntaxe

```
PREPARE statement_name FROM statement
```

Le tableau suivant décrit ces paramètres.

Paramètre	Description
statement_name	Nom de l'instruction à préparer. Le nom doit être unique au sein du groupe de travail.
déclaration	Une requête `SELECT`, `CTAS` ou `INSERT INTO`.

Exemple PREPARE

Les exemples suivants montrent l'utilisation de l'instruction `PREPARE`. Les points d'interrogation indiquent les valeurs à fournir par l'instruction `EXECUTE` lors de l'exécution de la requête.

```
PREPARE my_select1 FROM
```

```
SELECT * FROM nation
```

```
PREPARE my_select2 FROM
SELECT * FROM "my_database"."my_table" WHERE year = ?
```

```
PREPARE my_select3 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
```

```
PREPARE my_insert FROM
INSERT INTO cities_usa (city, state)
SELECT city, state
FROM cities_world
WHERE country = ?
```

```
PREPARE my_unload FROM
UNLOAD (SELECT * FROM table1 WHERE productid < ?)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format='PARQUET')
```

EXECUTE

Exécute une instruction préparée. Les valeurs des paramètres sont spécifiées dans la clause `USING`.

Syntaxe

```
EXECUTE statement_name [USING value1 [ ,value2, ... ] ]
```

`statement_name` est le nom de l'instruction préparée. *`value1`* et *`value2`* sont les valeurs à spécifier pour les paramètres de l'instruction.

Exemples EXECUTE

L'exemple suivant exécute l'instruction préparée `my_select1`, qui ne contient aucun paramètre.

```
EXECUTE my_select1
```

L'exemple suivant exécute l'instruction préparée `my_select2`, qui contient un seul paramètre.

```
EXECUTE my_select2 USING 2012
```

L'exemple suivant exécute l'instruction préparée `my_select3`, qui contient deux paramètres.

```
EXECUTE my_select3 USING 346078, 12
```

L'exemple suivant fournit une valeur de chaîne pour un paramètre dans l'instruction préparée `my_insert`.

```
EXECUTE my_insert USING 'usa'
```

L'exemple suivant fournit une valeur numérique pour le paramètre `productid` dans l'instruction préparée `my_unload`.

```
EXECUTE my_unload USING 12
```

DEALLOCATE PREPARE

Supprime l'instruction préparée portant le nom spécifié de la liste des instructions préparées dans le groupe de travail actuel.

Syntaxe

```
DEALLOCATE PREPARE statement_name
```

`statement_name` est le nom de l'instruction préparée à supprimer.

Exemple

L'exemple suivant supprime l'instruction préparée `my_select1` du groupe de travail actuel.

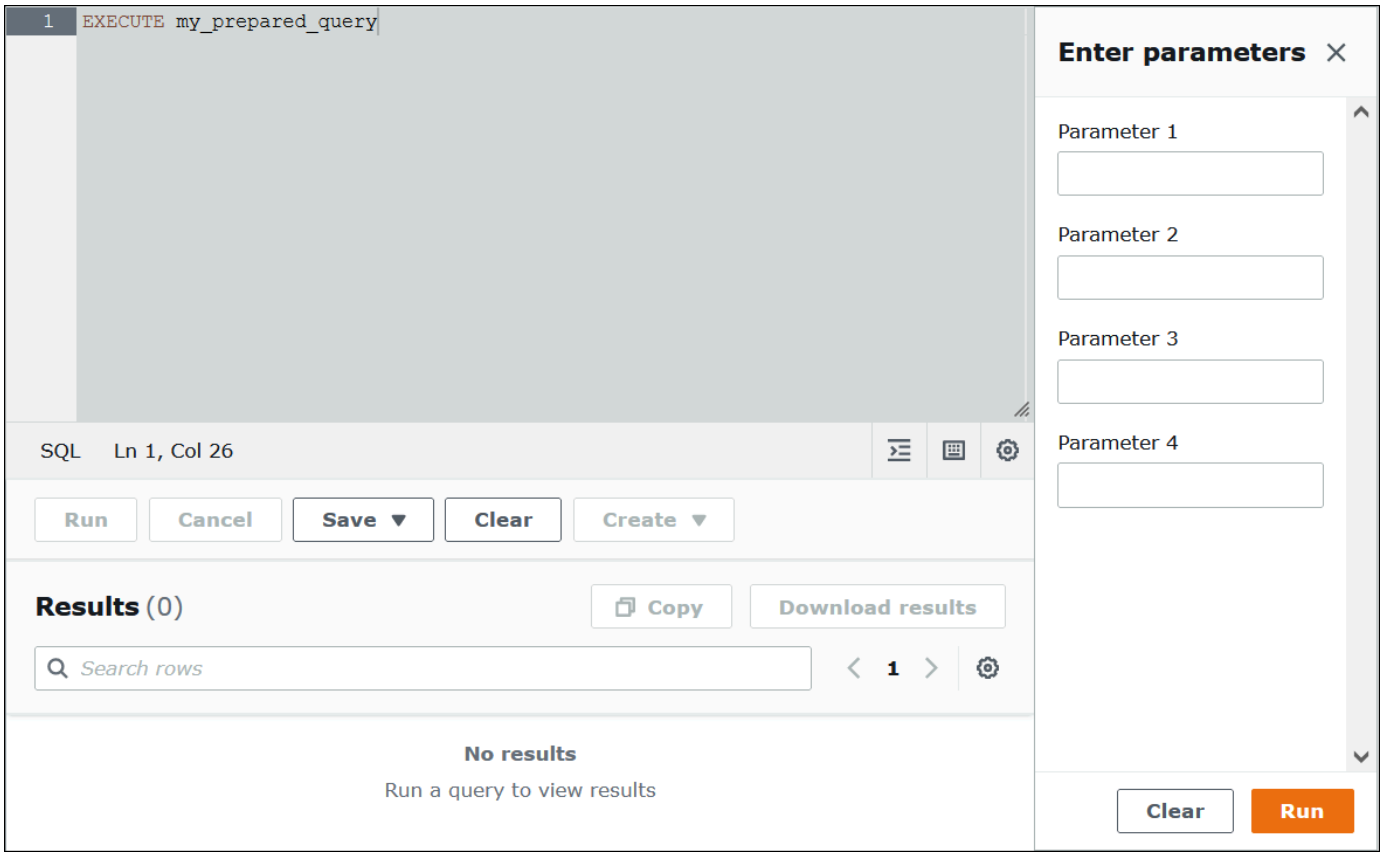
```
DEALLOCATE PREPARE my_select1
```

Exécution d'instructions préparées sans la clause USING dans la console Athena

Si vous exécutez une instruction préparée existante avec la syntaxe `EXECUTE` *`prepared_statement`* dans l'éditeur de requête, Athena ouvre la boîte de dialogue Enter parameters (Saisir des paramètres) pour que vous puissiez saisir les valeurs qui devraient normalement figurer dans la clause `USING` de l'instruction `EXECUTE ... USING`.

Pour exécuter une instruction préparée à l'aide de la boîte de dialogue Enter parameters (Saisir des paramètres)

1. Dans l'éditeur de requête, au lieu d'utiliser la syntaxe `EXECUTE prepared_statement USING` *`Value1`*`,` *`Value2`* `...`, utilisez plutôt la syntaxe `EXECUTE` *`prepared_statement`*.
2. Cliquez sur Exécuter. La boîte de dialogue Enter parameters (Saisir des paramètres)s'affiche.



3. Saisissez les valeurs dans l'ordre dans la boîte de dialogue Execution parameters (Paramètres d'exécution). Comme le texte d'origine de la requête n'est pas visible, vous devez vous souvenir de la signification de chaque paramètre positionnel ou disposer de l'instruction préparée pour référence.
4. Cliquez sur Exécuter.

Création de déclarations préparées à l'aide du AWS CLI

Pour utiliser le AWS CLI pour créer une instruction préparée, vous pouvez utiliser l'une des `athena` commandes suivantes :

- Utilisez la commande `create-prepared-statement` et fournissez une instruction de requête qui a des paramètres d'exécution.
- Utilisez la commande `start-query-execution` et fournissez une chaîne de requête qui utilise la syntaxe `PREPARE`.

En utilisant create-prepared-statement

Dans une commande `create-prepared-statement`, définissez le texte de la requête dans l'argument `query-statement`, comme dans l'exemple suivant.

```
aws athena create-prepared-statement
--statement-name PreparedStatement1
--query-statement "SELECT * FROM table WHERE x = ?"
--work-group athena-engine-v2
```

Utilisation start-query-execution et syntaxe PREPARE

Utilisez la commande `start-query-execution`. Placez l'instruction `PREPARE` dans l'argument `query-string`, comme dans l'exemple suivant :

```
aws athena start-query-execution
--query-string "PREPARE PreparedStatement1 FROM SELECT * FROM table WHERE x = ?"
--query-execution-context '{"Database": "default"}'
--result-configuration '{"OutputLocation": "s3://DOC-EXAMPLE-BUCKET/..."}'
```

Exécution d'instructions préparées à l'aide du AWS CLI

Pour exécuter une instruction préparée avec le AWS CLI, vous pouvez fournir des valeurs pour les paramètres en utilisant l'une des méthodes suivantes :

- Utilisez l'argument `execution-parameters`.
- Utilisez la syntaxe SQL `EXECUTE ... USING` dans l'argument `query-string`.

Utilisation de l'argument execution-parameters

Dans cette approche, vous utilisez la commande `start-query-execution` et saisissez le nom d'une instruction préparée existante dans l'argument `query-string`. Ensuite, dans l'argument `execution-parameters`, vous fournissez les valeurs des paramètres d'exécution. L'exemple suivant montre cette methode.

```
aws athena start-query-execution
--query-string "Execute PreparedStatement1"
--query-execution-context "Database"="default"
--result-configuration "OutputLocation"="s3://DOC-EXAMPLE-BUCKET/..."
--execution-parameters "1" "2"
```

Utilisation de EXECUTE ... Utilisation de la syntaxe SQL

Pour exécuter une instruction préparée existante à l'aide de la syntaxe `EXECUTE ... USING`, vous utilisez la commande `start-query-execution` et placez le nom de l'instruction préparée et les valeurs des paramètres dans l'argument `query-string`, comme dans l'exemple suivant :

```
aws athena start-query-execution
--query-string "EXECUTE PreparedStatement1 USING 1"
--query-execution-context '{"Database": "default"}'
--result-configuration '{"OutputLocation": "s3://DOC-EXAMPLE-BUCKET/..."}'
```

Listage d'instructions préparées

Pour répertorier les instructions préparées pour un groupe de travail spécifique, vous pouvez utiliser la commande list-prepared-statements AWS CLI Athena ou l'action ListPreparedStatementsAthena API. Le paramètre `--work-group` est obligatoire.

```
aws athena list-prepared-statements --work-group primary
```

Ressources supplémentaires

Consultez les articles connexes suivants sur le blog AWS Big Data.

- Améliorer la réutilisabilité et la sécurité grâce aux requêtes paramétrées Amazon Athena
- Utiliser des requêtes paramétrées Amazon Athena pour fournir des données en tant que service

Utilisation de l'optimiseur basé sur les coûts

Vous pouvez utiliser la fonctionnalité d'optimisation basée sur les coûts (CBO) d'Athena SQL pour optimiser vos requêtes. Vous pouvez éventuellement demander à Athena de recueillir des statistiques au niveau des tables ou des colonnes pour l'une de vos tables dans AWS Glue. Si toutes les tables de votre requête contiennent des statistiques, Athena utilise ces statistiques pour créer un

plan d'exécution qu'elle considère comme le plus performant. L'optimiseur de requêtes calcule des plans alternatifs sur la base d'un modèle statistique, puis sélectionne celui qui sera probablement le plus rapide pour exécuter la requête.

Les statistiques sur AWS Glue les tables sont collectées et stockées dans le AWS Glue Data Catalog et mises à la disposition d'Athena pour améliorer la planification et l'exécution des requêtes. Il s'agit de statistiques au niveau des colonnes, telles que le nombre de valeurs distinctes, le nombre de valeurs nulles, maximales et minimales pour des types de fichiers tels que Parquet, ORC, JSON, ION, CSV et XML. Amazon Athena utilise ces statistiques pour optimiser les requêtes en appliquant les filtres les plus restrictifs le plus tôt possible lors du traitement des requêtes. Ce filtrage limite l'utilisation de la mémoire et le nombre d'enregistrements qui doivent être lus pour fournir les résultats de la requête.

En conjonction avec le CBO, Athena utilise une fonctionnalité appelée optimiseur basé sur des règles (RBO). Le RBO applique mécaniquement des règles censées améliorer les performances des requêtes. Le RBO est généralement bénéfique, car ses transformations visent à simplifier le plan de requêtes. Cependant, étant donné que RBO n'effectue pas de calculs de coûts ni de comparaisons de plans, il lui est difficile de créer un plan optimal à partir de requêtes plus complexes.

C'est pourquoi Athena utilise à la fois le RBO et le CBO pour optimiser vos requêtes. Après avoir identifié les possibilités d'améliorer l'exécution des requêtes, Athena crée un plan optimal. Pour en savoir plus sur les détails du plan d'exécution, veuillez consulter Affichage des plans d'exécution pour les requêtes SQL. Pour une discussion détaillée du fonctionnement du CBO, consultez l'article Accélérer les requêtes avec l'optimiseur basé sur les coûts d'Amazon Athena sur le blog Big Data. AWS

Pour générer des statistiques pour les tables du AWS Glue catalogue, vous pouvez utiliser la console Athena, la AWS Glue console ou AWS Glue les API. Athena étant intégrée à AWS Glue Catalog, vous bénéficiez automatiquement des améliorations de performances correspondantes lorsque vous exécutez des requêtes depuis Amazon Athena.

Considérations et restrictions

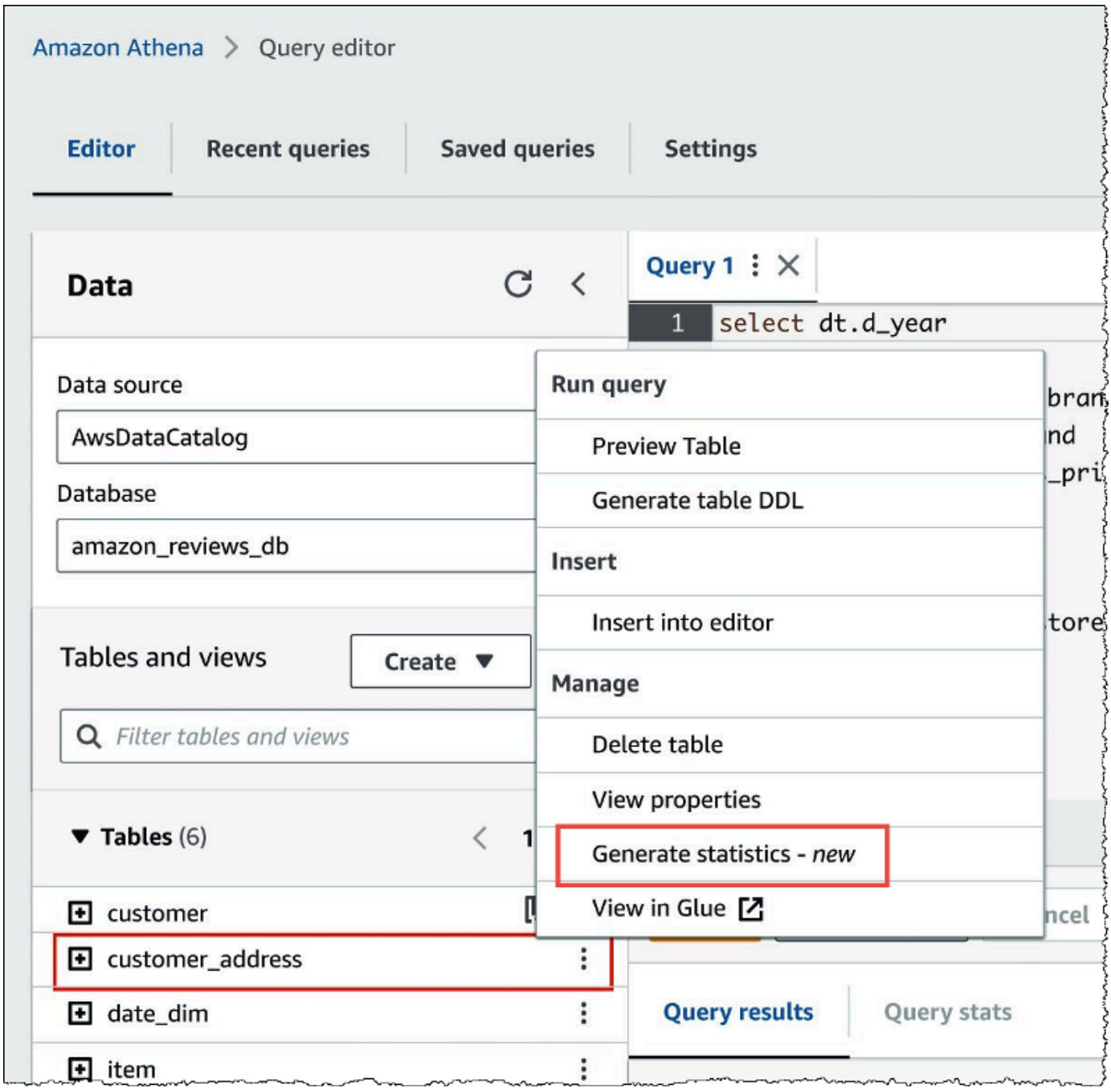
- Types de tables : actuellement, la fonctionnalité CBO d'Athena ne prend en charge que les tables Hive qui se trouvent dans l' AWS Glue Data Catalog.
- Athena pour Spark : la fonctionnalité CBO n'est pas disponible dans Athena pour Spark.
- Tarification : pour obtenir des informations sur les prix, consultez la page de tarification d'AWS Glue.

Génération de statistiques sur les tables à l'aide de la console Athena

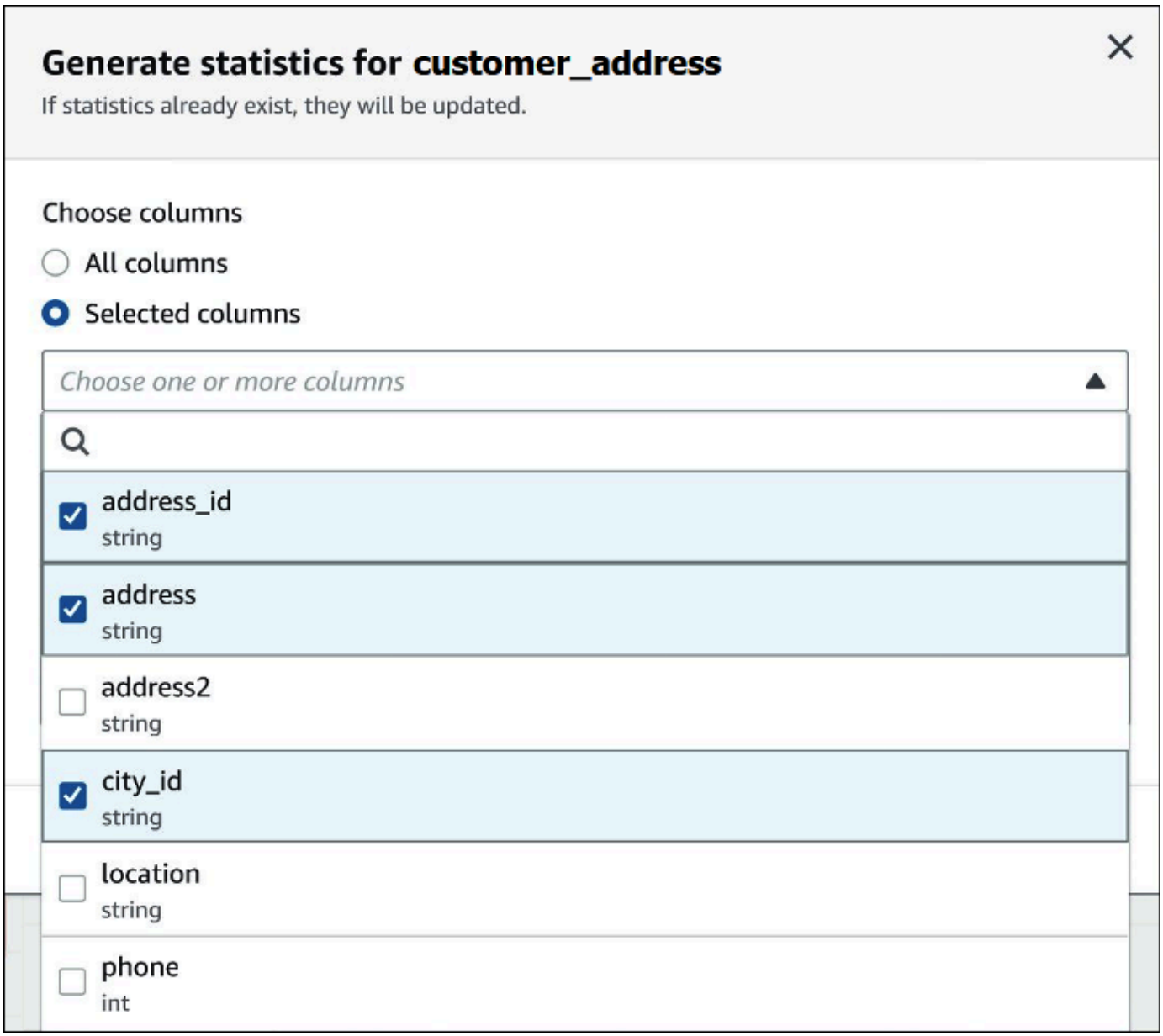
Cette section explique comment utiliser la console Athena pour générer des statistiques au niveau des tables ou des colonnes pour une table dans AWS Glue. Pour plus d'informations sur l'utilisation AWS Glue pour générer des statistiques de table, consultez la section Utilisation des statistiques de colonnes dans le Guide du AWS Glue développeur.

Pour générer des statistiques sur les tables à l'aide de la console Athena

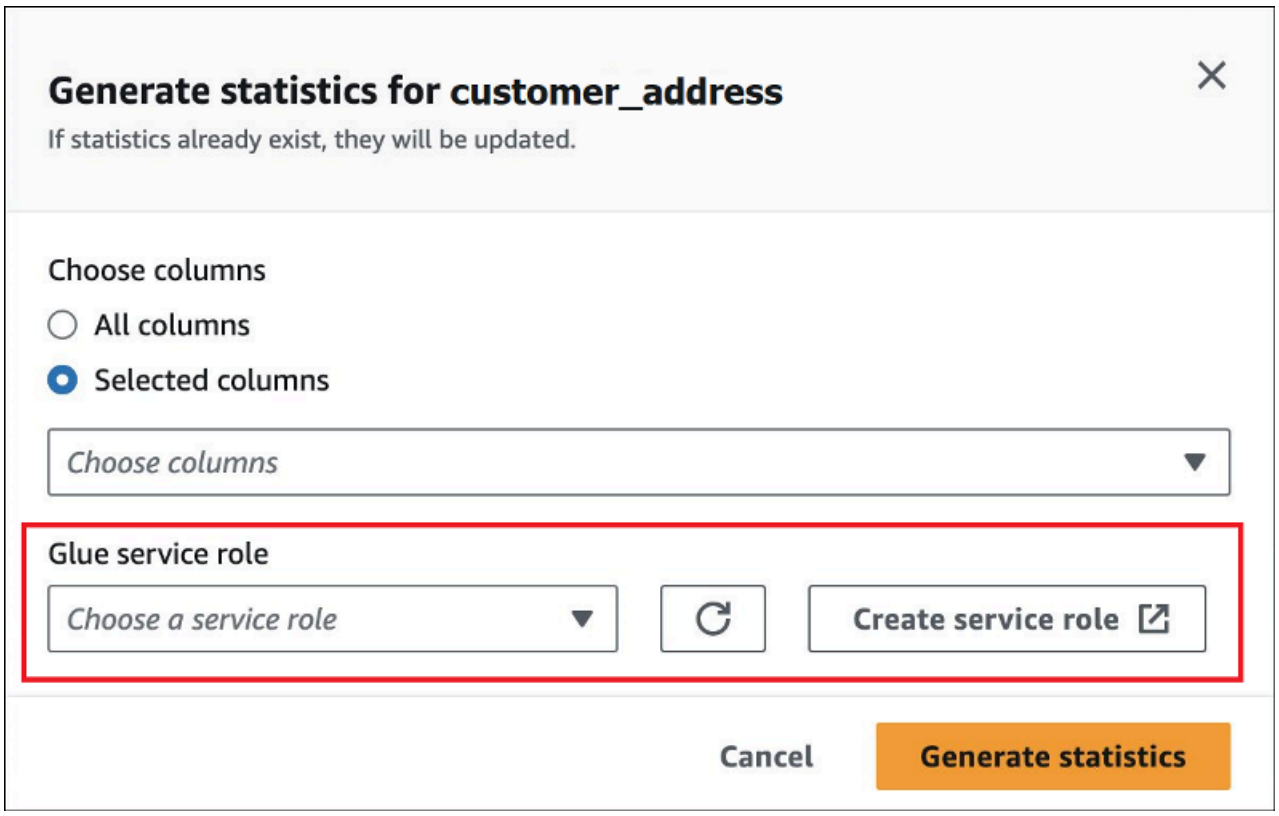
1. Ouvrez la console Athena à l’adresse https://console.aws.amazon.com/athena/.
2. Dans la liste des Tables de l'éditeur de requêtes Athena, choisissez les trois points verticaux pour le tableau de votre choix, puis sélectionnez Générer des statistiques.



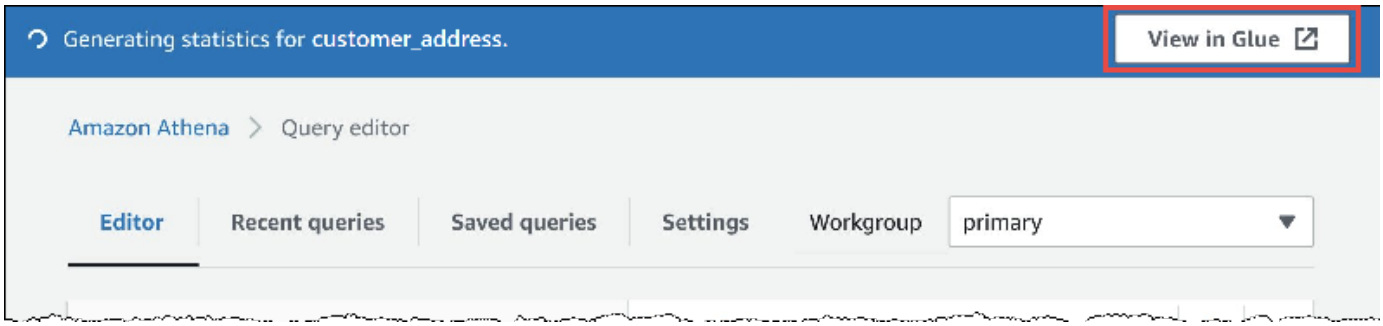
3. Dans la boîte de dialogue Générer des statistiques, choisissez Toutes les colonnes pour générer des statistiques pour toutes les colonnes de la table, ou sélectionnez Colonnes sélectionnées pour sélectionner des colonnes spécifiques. La valeur par défaut est Toutes les colonnes.



4. Pour le rôle de AWS Glue service, créez ou sélectionnez un rôle de service existant AWS Glue pour autoriser la génération de statistiques. La fonction du service AWS Glue nécessite également des autorisations `S3:GetObject` d'accès au compartiment Amazon S3 qui contient les données de la table.



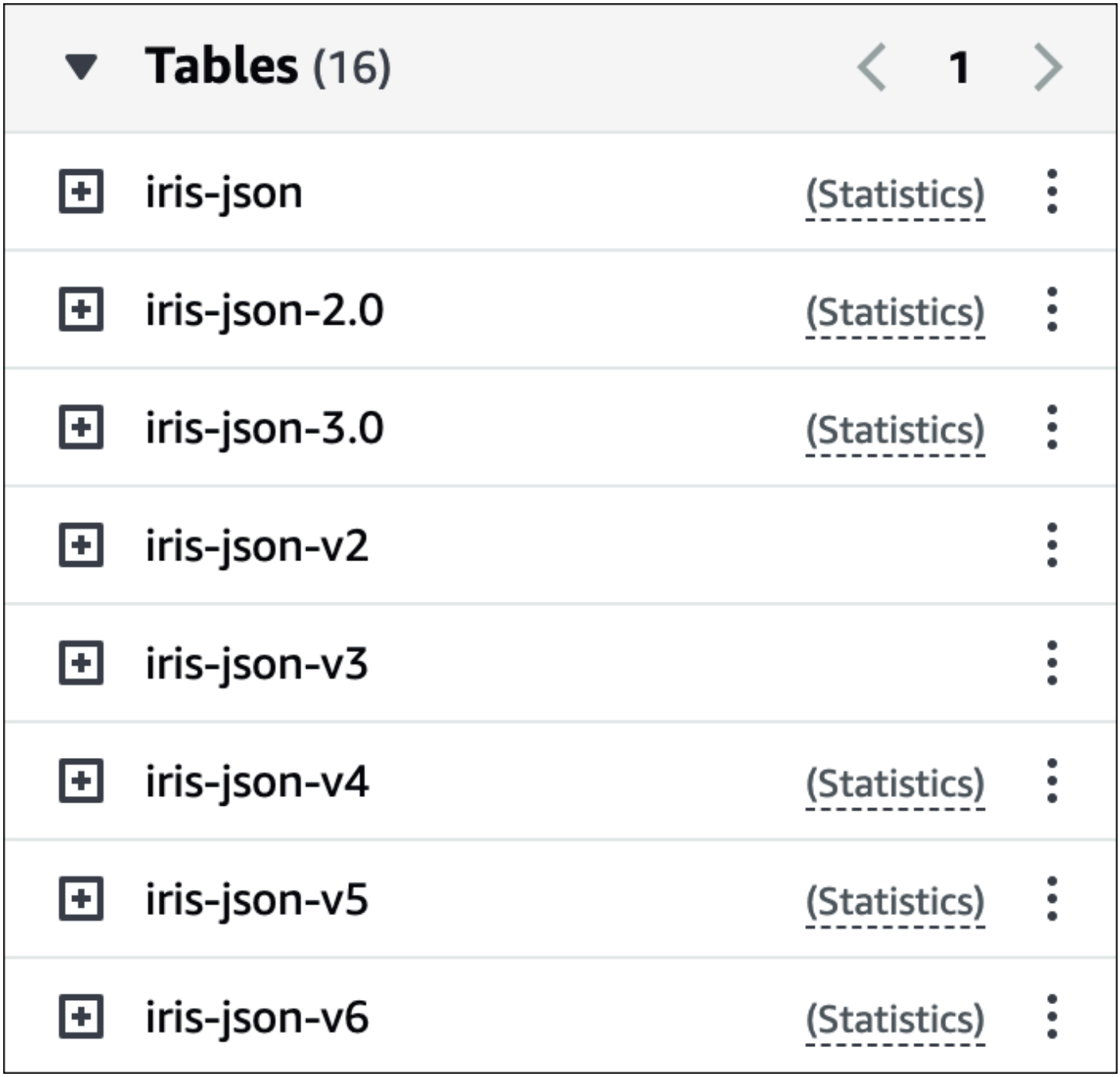
5. Choisissez Générer des statistiques. Une bannière de notification Génération de statistiques pour ***`table_name`*** affiche le statut de la tâche.



6. Pour afficher les détails dans la AWS Glue console, choisissez View in Glue.

   Pour plus d'informations sur l'affichage des statistiques dans la AWS Glue console, consultez la section Affichage des statistiques des colonnes dans le Guide du AWS Glue développeur.

7. Une fois les statistiques générées, les tables et les colonnes contenant des statistiques affichent le mot Statistiques entre parenthèses, comme dans l'image suivante.



Désormais, lorsque vous exécutez vos requêtes, Athena effectue une optimisation basée sur les coûts sur les tables et les colonnes pour lesquelles les statistiques ont été générées.

Ressources supplémentaires

Pour plus d'informations, consultez la ressource suivante.

Interrogation des données de S3 Express One Zone

La classe de stockage Amazon S3 Express One Zone est une classe de stockage Amazon S3 très performante qui fournit des temps de réponse de l'ordre de la milliseconde. En tant que tel, il est utile pour les applications qui accèdent fréquemment aux données avec des centaines de milliers de requêtes par seconde.

S3 Express One Zone réplique et stocke les données au sein de la même zone de disponibilité afin d'optimiser la vitesse et les coûts. Cela diffère des classes de stockage régionales Amazon S3, qui répliquent automatiquement les données sur au moins trois zones de AWS disponibilité au sein d'un Région AWS.

Pour plus d'informations, veuillez consulter What is S3 Express One Zone? dans le Guide de l'utilisateur Amazon S3.

Prérequis

Vérifiez que les conditions suivantes sont réunies avant de commencer :

- Moteur Athena version 3 : pour utiliser S3 Express One Zone avec Athena SQL, votre groupe de travail doit être configuré pour utiliser le moteur Athena version 3.
- Autorisations S3 Express One Zone : lorsque S3 Express One Zone appelle une action telle que `GET`, `LIST` ou `PUT` sur un objet Amazon S3, la classe de stockage appelle `CreateSession` en votre nom. Pour cette raison, votre politique IAM doit autoriser l'action `s3express:CreateSession`, qui permet à Athena d'invoquer l'opération d'API correspondante.

Considérations et restrictions

Lorsque vous interrogez S3 Express One Zone avec Athena, tenez compte des points suivants.

- Les compartiments S3 Express One Zone ne prennent en charge que le chiffrement `SSE_S3`. Les résultats des requêtes Athena sont écrits à l'aide du chiffrement `SSE_S3`, quelle que soit l'option que vous avez choisie dans les paramètres du groupe de travail pour chiffrer les résultats des requêtes. Cette limitation inclut tous les scénarios dans lesquels Athena écrit des données dans des compartiments S3 Express One Zone, y compris les instructions `CREATE TABLE AS` (CTAS) et `INSERT INTO`.
- Le AWS Glue robot d'exploration n'est pas pris en charge pour créer des tables sur les données S3 Express One Zone.

- L'instruction `MSCK REPAIR TABLE` n'est pas prise en charge. Comme solution de contournement, utilisez ALTER TABLE ADD PARTITION.
- `ALTER TABLE ADD PARTITIONALTER TABLE DROP PARTITION`, et ne `ALTER TABLE RENAME PARTITION` sont pas pris en charge pour les tables Iceberg dans S3 Express One Zone.
- Les formats de fichiers et de tables suivants ne sont pas pris en charge ou sont pris en charge de manière limitée. Si des formats ne sont pas répertoriés, mais qu'ils sont pris en charge par Athena (comme Parquet, ORC et JSON), ils sont également pris en charge avec le stockage S3 Express One Zone.

Format de fichier ou de table	Limitation
Apache Avro	Non pris en charge
CloudTrail journaux	Non pris en charge
Apache Hudi	Non pris en charge
Amazon Ion	Non pris en charge
Journaux Logstash	Non pris en charge
WebServer Journaux Apache	Non pris en charge
Delta Lake	DDL non prise en charge. Pour plus d'informations sur la création d'une table Delta Lake à l'aide d'un schéma factice, consultez Synchronisation des métadonnées Delta Lake. Les requêtes `SELECT` sur la table sont prises en charge.

Premiers pas

L'interrogation des données S3 Express One Zone avec Athena est simple. Pour démarrer, suivez la procédure ci-dessous.

Pour utiliser Athena SQL afin d'interroger les données S3 Express One Zone

1. Transférez vos données vers le stockage S3 Express One Zone. Pour de plus amples informations, consultez la rubrique Définition de la classe de stockage d'un objet dans le Guide de l'utilisateur Amazon S3.
2. Utilisez une instruction CREATE TABLE dans Athena pour cataloguer vos données dans AWS Glue Data Catalog. Pour plus d'informations sur la création des tables dans Athena, consultez Création de tables dans Athena et l'instruction CREATE TABLE.
3. (Facultatif) Configurez l'emplacement des résultats de requête de votre groupe de travail Athena pour utiliser un compartiment de répertoire Amazon S3. Les compartiments de répertoire Amazon S3 sont plus performants que les compartiments généraux et sont conçus pour les charges de travail ou les applications critiques en termes de performances qui nécessitent une latence constante de l'ordre de la milliseconde. Pour plus d'informations, consultez la rubrique Directory buckets overview dans le Guide de l'utilisateur Amazon S3.

Interrogation d'objets Amazon S3 Glacier restaurés

Vous pouvez utiliser Athena pour interroger des objets restaurés à partir des classes de stockage Amazon S3 S3 Glacier Flexible Retrieval (anciennement Glacier) et S3 Glacier Deep Archive. Vous devez activer cette fonctionnalité par table. Si vous ne souhaitez pas activer la fonctionnalité sur une table avant d'exécuter une requête, Athena ignore tous les objets S3 Glacier Flexible Retrieval et S3 Glacier Deep Archive de la table lors de l'exécution de la requête.

Considérations et restrictions

- L'interrogation des objets Amazon S3 Glacier restaurés est prise en charge uniquement sur la version 3 du moteur Athena.
- La fonctionnalité est prise en charge uniquement pour les tables Apache Hive.
- Vous devez restaurer vos objets avant d'interroger vos données ; Athena ne les restaure pas à votre place.

Configuration d'une table pour utiliser les objets restaurés

Pour configurer votre table Athena afin d'inclure des objets restaurés dans vos requêtes, vous devez définir sa propriété de table `read_restored_glacier_objects` sur `true`. Pour ce faire, vous

pouvez utiliser l'éditeur de requêtes Athena ou la AWS Glue console. Vous pouvez également utiliser l'interface de ligne de commande AWS Glue, l'API AWS Glue ou le kit SDK AWS Glue.

Utilisation de l'éditeur de requêtes Athena

Dans Athena, vous pouvez utiliser la commande ALTER TABLE SET TBLPROPERTIES pour définir la propriété de table, comme dans l'exemple suivant.

```
ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'true')
```

Utilisation de la console AWS Glue

Dans la AWS Glue console, effectuez les étapes suivantes pour ajouter la propriété de `read_restored_glacier_objects` table.

Pour configurer les propriétés du tableau dans la AWS Glue console

1. Connectez-vous à la AWS Glue console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/glue/.
2. Effectuez l'une des actions suivantes :
   - Choisissez Accéder au catalogue de données.
   - Dans le panneau de navigation, choisissez Tables du catalogue de données.
3. Sur la page Tables, dans la liste des tables, choisissez le lien correspondant à la table que vous souhaitez modifier.
4. Sélectionnez Actions, puis Modifier la table.
5. Sur la page Modifier la table, dans la section Propriétés de la table, ajoutez la paire clé-valeur suivante.
   - Dans Clé, ajoutez `read_restored_glacier_objects`.
   - Pour le champ Value (Valeur), entrez `true`.
6. Choisissez Enregistrer.

À l'aide du AWS CLI

Dans le AWS CLI, vous pouvez utiliser la commande AWS Glue update-table et son `--table-input` argument pour redéfinir la table et, ce faisant, ajouter la propriété.

`read_restored_glacier_objects` Dans l'argument `--table-input`, utilisez la structure `Parameters` pour spécifier la propriété `read_restored_glacier_objects` et la valeur de `true`. Notez que l'argument pour `--table-input` ne doit pas comporter d'espaces et doit utiliser des barres obliques inverses pour éviter les guillemets doubles. Dans l'exemple suivant, remplacez *`my_database`* et *`my_table`* par les noms de votre base de données et de votre table.

```
aws glue update-table \
   --database-name my_database \
   --table-input={\"Name\":\"my_table\",\"Parameters\":{\"read_restored_glacier_objects
\":\"true\"}}
```

> **Important**
>
> La AWS Glue `update-table` commande fonctionne en mode de remplacement, ce qui signifie qu'elle remplace la définition de table existante par la nouvelle définition spécifiée par le `table-input` paramètre. Pour cette raison, veillez également à spécifier tous les champs que vous souhaitez voir figurer dans votre table dans le `table-input` paramètre lorsque vous ajoutez la `read_restored_glacier_objects` propriété.

Traitement des mises à jour de schéma

Cette section fournit des conseils sur le traitement des mises à jour de schémas pour divers formats de données. Athena est un moteur de schema-on-read requêtes. Cela signifie que lorsque vous créez une table dans Athena, elle s'applique aux schémas lors de la lecture des données. Elle ne modifie ni ne réécrit les données sous-jacentes.

Si vous anticipez des modifications dans les schémas de table, envisagez de les créer dans un format de données qui est adapté à vos besoins. Vos objectifs sont de réutiliser les requêtes Athena existantes dans des schémas qui évoluent et d'éviter les erreurs de concordance de schéma lors de requêtes sur des tables avec des partitions.

Pour atteindre ces objectifs, choisissez un format de données pour la table en fonction de celle-ci dans la rubrique suivante.

Rubriques

- Résumé : Mises à jour et formats de données dans Athena
- Accès à l'index dans parquet et ORC

- Types de mises à jour
- Mises à jour dans les tables avec des partitions

Résumé : Mises à jour et formats de données dans Athena

Le tableau suivant récapitule les formats de stockage de données et les manipulations de schéma prises en charge correspondantes. Utilisez ce tableau pour vous aider à choisir le format qui vous permettra de continuer à utiliser les requêtes Athena, même lorsque vos schémas évoluent au fil du temps.

Dans ce tableau, notez que Parquet et ORC sont des formats de données en colonnes avec des méthodes d'accès aux colonnes par défaut différentes. Par défaut, Parquet accède aux colonnes par nom et ORC par index (valeur ordinale). Athena fournit donc une SerDe propriété définie lors de la création d'une table pour activer la méthode d'accès aux colonnes par défaut, ce qui permet une plus grande flexibilité lors de l'évolution du schéma.

Pour Parquet, la propriété `parquet.column.index.access` peut être définie sur `true`, ce qui définit la méthode d'accès aux colonnes pour utiliser le numéro ordinal de la colonne. La définition de cette propriété sur `false` change la méthode d'accès aux colonnes pour utiliser le nom de colonne. De même, pour ORC, utilisez la propriété `orc.column.index.access` pour contrôler la méthode d'accès aux colonnes. Pour plus d’informations, consultez Accès à l'index dans parquet et ORC.

CSV et TSV vous permettent d'effectuer toutes les manipulations de schéma, sauf réorganiser les colonnes ou ajouter des colonnes au début de la table. Par exemple, si l'évolution de votre schéma nécessite uniquement que vous renommiez des colonnes, sans les supprimer, vous pouvez choisir de créer vos tables au format CSV ou TSV. Si vous avez besoin de supprimer des colonnes, n'utilisez pas CSV ou TSV. Utilisez plutôt l'un des autres formats pris en charge, de préférence un format en colonnes, comme Parquet ou ORC.

Mises à jour de schéma et formats de données dans Athena

Type de mise à jour de schéma attendue	Récapitulatif	CSV (avec et sans en-têtes) et TSV	JSO	AVF	PARQUE Lecture par nom (par défaut)	PARQUE Lecture par index	ORC : Lecture par index (par défaut)	ORC : Lecture par nom
Changement de nom de colonne	Stockez vos données aux formats CSV et TSV, ou aux formats Parquet et ORC ou si elles sont lues par index.	Y	N	N	N	Y	Y	N
Ajouter des colonnes au début ou au milieu de la table	Stockez vos données aux formats JSON, AVRO, ou aux formats ORC et Parquet si elles sont lues par nom. N'utilisez pas CSV et TSV.	N	Y	Y	Y	N	N	Y
Ajouter des colonnes à la fin de la table	Stockez vos données aux formats CSV ou TSV, JSON, ORC, AVRO ou Parquet.	Y	Y	Y	Y	Y	Y	Y

Type de mise à jour de schéma attendue	Récapitulatif	CSV (avec et sans en-têtes) et TSV	JSO	AVI	PARQUE Lecture par nom (par défaut)	PARQUE Lecture par index	ORC : Lecture par index (par défaut)	ORC : Lecture par nom
Supprimer des colonnes	Stockez vos données aux formats JSON, AVRO, ou aux formats Parquet et ORC si elles sont lues par nom. N'utilisez pas CSV et TSV.	N	Y	Y	Y	N	N	Y
Réorganiser des colonnes	Stockez vos données aux formats AVRO, JSON, ou aux formats Parquet et ORC si elles sont lues par nom.	N	Y	Y	Y	N	N	Y

Type de mise à jour de schéma attendue	Récapitulatif	CSV (avec et sans en-têtes) et TSV	JSO	AVF	PARQUE Lecture par nom (par défaut)	PARQUE Lecture par index	ORC : Lecture par index (par défaut)	ORC : Lecture par nom
Modifier le type de données d'une colonne	Stockez vos données dans n'importe quel format, mais testez votre requête dans Athena pour vous assurer que les types de données sont compatibles. Pour Parquet et ORC, la modification d'un type de données fonctionne uniquement pour les tables partitionnées.	Y	Y	Y	Y	Y	Y	Y

Accès à l'index dans parquet et ORC

PARQUET et ORC sont des formats de stockage de données en colonnes pouvant être lus par index ou par nom. Le stockage de vos données dans l'un ou l'autre de ces formats vous permet d'effectuer toutes les opérations sur les schémas et d'exécuter des requêtes Athena sans erreur de non-concordance de schéma.

- Athena lit ORC par index par défaut, comme défini dans `SERDEPROPERTIES ( 'orc.column.index.access'='true')`. Pour plus d'informations, consultez ORC : Lecture par index.
- Athena lit Parquet par nom par défaut, comme défini dans `SERDEPROPERTIES ( 'parquet.column.index.access'='false')`. Pour plus d'informations, consultez Parquet : Lecture par nom.

Comme il s'agit de valeurs par défaut, la spécification de ces SerDe propriétés dans vos `CREATE TABLE` requêtes est facultative, elles sont utilisées implicitement. Lorsqu'elles sont utilisées, elles vous permettent d'exécuter certaines opérations de mise à jour de schéma tout en empêchant d'autres opérations. Pour activer ces opérations, exécutez une autre `CREATE TABLE` requête et modifiez les SerDe paramètres.

> **Note**
>
> Les SerDe propriétés ne sont pas automatiquement propagées à chaque partition. Utilisez `ALTER TABLE ADD PARTITION` des instructions pour définir les SerDe propriétés de chaque partition. Pour automatiser ce processus, écrivez un script qui exécute les instructions `ALTER TABLE ADD PARTITION`.

Les sections suivantes décrivent ces cas en détail.

ORC : Lecture par index

Une table au format ORC est lue par index, par défaut. Ce comportement est défini par la syntaxe suivante :

```
WITH SERDEPROPERTIES (
  'orc.column.index.access'='true')
```

La lecture par index vous permet de renommer des colonnes. Mais vous perdez alors la possibilité de supprimer des colonnes ou d'en ajouter au milieu de la table.

Pour que ORC soit lu par son nom, ce qui vous permettra d'ajouter des colonnes au milieu du tableau ou de supprimer des colonnes dans ORC, définissez la SerDe propriété sur `false` dans `orc.column.index.access` l'`CREATE TABLE`instruction. Dans cette configuration, vous perdrez la possibilité de renommer des colonnes.

> **Note**
>
> Dans la version 2 du moteur Athena, lorsque les tables ORC sont configurées pour être lues par nom, Athena exige que tous les noms de colonnes dans les fichiers ORC soient en minuscules. Comme Apache Spark ne met pas en minuscules les noms de champs lorsqu'il génère des fichiers ORC, Athena risque de ne pas pouvoir lire les données ainsi générées. La solution consiste à renommer les colonnes pour qu'elles soient en minuscules ou à utiliser la version 3 du moteur Athena.

L'exemple suivant montre comment modifier le format ORC pour qu'il soit lu par nom :

```
CREATE EXTERNAL TABLE orders_orc_read_by_name (
   `o_comment` string,
   `o_orderkey` int,
   `o_custkey` int,
   `o_orderpriority` string,
   `o_orderstatus` string,
   `o_clerk` string,
   `o_shippriority` int,
   `o_orderdate` string
)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
WITH SERDEPROPERTIES (
  'orc.column.index.access'='false')
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_orc/';
```

Parquet : Lecture par nom

Une table au format Parquet est lue par nom, par défaut. Ce comportement est défini par la syntaxe suivante :

```
WITH SERDEPROPERTIES (
  'parquet.column.index.access'='false')
```

La lecture par nom vous permet d'ajouter des colonnes au milieu de la table et de supprimer des colonnes. Mais vous perdez alors la possibilité de renommer des colonnes.

Pour que Parquet soit lu par index, ce qui vous permettra de renommer les colonnes, vous devez créer une table dont la `parquet.column.index.access` SerDe propriété est définie sur. `true`

Types de mises à jour

Cette rubrique décrit certaines des modifications que vous pouvez apporter au schéma dans les instructions `CREATE TABLE` sans réellement modifier vos données. Nous passons en revue chaque type de mise à jour de schéma et précisons quels formats de données les permettent dans Athena. Pour mettre à jour un schéma, vous pouvez dans certains cas utiliser une commande `ALTER TABLE`, mais dans d'autres cas, vous ne modifiez pas réellement une table existante. À la place, vous créez une table avec un nouveau nom qui modifie le schéma que vous avez utilisé dans votre instruction `CREATE TABLE` d'origine.

- Ajout de colonnes au début ou au milieu de la table
- Ajout de colonnes à la fin de la table
- Suppression de colonnes
- Changement de nom de colonnes
- Réorganisation des colonnes
- Modification du type de données d'une colonne

En fonction de l'évolution attendue de vos schémas, pour continuer à utiliser les requêtes Athena, choisissez un format de données compatible.

Prenons une application qui lit les informations de commandes à partir d'une table `orders` qui existe dans deux formats : CSV et Parquet.

L'exemple suivant crée une table au format Parquet :

```
CREATE EXTERNAL TABLE orders_parquet (
   `orderkey` int,
   `orderstatus` string,
   `totalprice` double,
   `orderdate` string,
   `orderpriority` string,
   `clerk` string,
   `shippriority` int
```

```
) STORED AS PARQUET
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_ parquet/';
```

L'exemple suivant crée cette même table au format CSV :

```
CREATE EXTERNAL TABLE orders_csv (
   `orderkey` int,
   `orderstatus` string,
   `totalprice` double,
   `orderdate` string,
   `orderpriority` string,
   `clerk` string,
   `shippriority` int
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_csv/';
```

Dans les sections suivantes, nous verrons dans quelle mesure les mises à jour apportées à ces tables affectent les requêtes Athena.

Ajout de colonnes au début ou au milieu de la table

L'ajout de colonnes est l'une des modifications de schéma les plus fréquentes. Par exemple, vous pouvez ajouter une nouvelle colonne pour enrichir la table avec de nouvelles données. Ou bien, vous pouvez ajouter une nouvelle colonne si la source d'une colonne existante a changé, et conserver la version précédente de cette colonne, pour ajuster les applications qui en dépendent.

Pour ajouter des colonnes au début ou au milieu de la table et continuer à exécuter des requêtes sur des tables existantes, utilisez AVRO, JSON, Parquet et ORC si leur SerDe propriété est définie pour lire par nom. Pour plus d'informations, veuillez consulter Accès à l'index dans parquet et ORC.

N'ajoutez pas colonnes au début ou au milieu de la table dans CSV et TSV, car ces formats dépendent de l'ordre. L'ajout d'une colonne dans de tels cas provoquera des erreurs de non-concordance de schéma en cas de modification du schéma de partitions.

L'exemple suivant crée une nouvelle table qui ajoute une colonne `o_comment` au milieu d'une table basée sur des données JSON.

```
CREATE EXTERNAL TABLE orders_json_column_addition (
   `o_orderkey` int,
   `o_custkey` int,
```

```
    `o_orderstatus` string,
    `o_comment` string,
    `o_totalprice` double,
    `o_orderdate` string,
    `o_orderpriority` string,
    `o_clerk` string,
    `o_shippriority` int,
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_json/';
```

Ajout de colonnes à la fin de la table

Si vous créez des tables dans l'un des formats pris en charge par Athena, tels que Parquet, ORC, Avro, JSON, CSV et TSV, vous pouvez utiliser l'instruction `ALTER TABLE ADD COLUMNS` pour ajouter des colonnes après les colonnes existantes, mais avant les colonnes de partition.

L'exemple suivant ajoute une colonne `comment` à la fin de la table `orders_parquet` avant les colonnes de partition :

```
ALTER TABLE orders_parquet ADD COLUMNS (comment string)
```

> **Note**
>
> Pour afficher une nouvelle colonne de table dans l'éditeur de requête après l'exécution de `ALTER TABLE ADD COLUMNS`, actualisez manuellement la liste des tables dans l'éditeur, puis développez à nouveau la table.

Suppression de colonnes

Vous devrez peut-être supprimer des colonnes dans des tables si elles ne contiennent plus de données, ou pour restreindre l'accès aux données qu'elles contiennent.

- Vous pouvez supprimer des colonnes de tables aux formats JSON, Avro, et aux formats Parquet et ORC s'ils sont lus par nom. Pour plus d'informations, veuillez consulter Accès à l'index dans parquet et ORC.
- Il est déconseillé de supprimer des colonnes de tables aux formats CSV et TSV si vous souhaitez conserver les tables que vous avez déjà créées dans Athena. La suppression d'une colonne casse le schéma et exige que vous recréiez la table sans la colonne supprimée.

Dans cet exemple, vous supprimez une colonne `totalprice` d'une table dans Parquet et exécuter une requête. Dans Athena, Parquet est lu par le nom par défaut ; c'est pourquoi nous omettons la configuration SERDEPROPERTIES qui spécifie la lecture par nom. Notez que la requête suivante réussit, même si vous avez modifié le schéma :

```
CREATE EXTERNAL TABLE orders_parquet_column_removed (
   `o_orderkey` int,
   `o_custkey` int,
   `o_orderstatus` string,
   `o_orderdate` string,
   `o_orderpriority` string,
   `o_clerk` string,
   `o_shippriority` int,
   `o_comment` string
)
STORED AS PARQUET
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_parquet/';
```

Changement de nom de colonnes

Vous pouvez, si vous le souhaitez, renommer les colonnes dans vos tables pour corriger l'orthographe, créer des noms de colonnes plus descriptifs, ou encore réutiliser une colonne existante afin d'éviter d'avoir à réorganiser les colonnes.

Vous pouvez renommer des colonnes si vous stockez vos données aux formats CSV et TSV, ou aux formats Parquet et ORC qui sont configurés pour lire par index. Pour plus d'informations, veuillez consulter Accès à l'index dans parquet et ORC.

Athena lit les données au format CSV et TSV dans l'ordre des colonnes du schéma et les renvoie dans le même ordre. Il n'utilise pas les noms de colonnes pour mapper des données à une colonne. C'est pourquoi vous ne pouvez pas renommer les colonnes dans CSV ou TSV sans interrompre les requêtes Athena.

Une politique pour renommer les colonnes consiste à créer une nouvelle table basée sur les mêmes données sous-jacentes, mais en utilisant de nouveaux noms de colonnes. L'exemple suivant crée une nouvelle table `orders_parquet` appelée `orders_parquet_column_renamed`. L'exemple change le nom `o_totalprice` de la colonne en `o_total_price` et exécute ensuite une requête dans Athena :

```
CREATE EXTERNAL TABLE orders_parquet_column_renamed (
```

```
   `o_orderkey` int,
   `o_custkey` int,
   `o_orderstatus` string,
   `o_total_price` double,
   `o_orderdate` string,
   `o_orderpriority` string,
   `o_clerk` string,
   `o_shippriority` int,
   `o_comment` string
)
STORED AS PARQUET
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_parquet/';
```

Dans le cas de la table Parquet, la requête suivante s'exécute, mais la colonne renommée n'affiche pas de données, car l'accès à la colonne s'est fait par nom (par défaut dans Parquet) et non par index :

```
SELECT *
FROM orders_parquet_column_renamed;
```

Une requête avec une table dans CSV est similaire:

```
CREATE EXTERNAL TABLE orders_csv_column_renamed (
   `o_orderkey` int,
   `o_custkey` int,
   `o_orderstatus` string,
   `o_total_price` double,
   `o_orderdate` string,
   `o_orderpriority` string,
   `o_clerk` string,
   `o_shippriority` int,
   `o_comment` string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_csv/';
```

Dans le cas de la table CSV, la requête suivante s'exécute et les données s'affichent dans toutes les colonnes, y compris celle qui a été renommée:

```
SELECT *
FROM orders_csv_column_renamed;
```

Réorganisation des colonnes

Vous pouvez réorganiser les colonnes uniquement pour des tables avec des données dans des formats lus par nom, comme JSON ou Parquet, qui lisent par nom par défaut. Vous pouvez également faire lire le format ORC par nom, si nécessaire. Pour plus d'informations, veuillez consulter Accès à l'index dans parquet et ORC.

L'exemple suivant crée une nouvelle table avec les colonnes dans un ordre différent :

```
CREATE EXTERNAL TABLE orders_parquet_columns_reordered (
   `o_comment` string,
   `o_orderkey` int,
   `o_custkey` int,
   `o_orderpriority` string,
   `o_orderstatus` string,
   `o_clerk` string,
   `o_shippriority` int,
   `o_orderdate` string
)
STORED AS PARQUET
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_parquet/';
```

Modification du type de données d'une colonne

Vous souhaiterez peut-être utiliser un autre type de colonne lorsque le type existant ne peut plus contenir la quantité d'informations requise. Par exemple, les valeurs d'une colonne ID peuvent dépasser la taille du type de données `INT` et nécessiter l'utilisation du type de données `BIGINT`.

Lorsque vous envisagez d'utiliser un autre type de données pour une colonne, tenez compte des points suivants :

- Dans la plupart des cas, vous ne pouvez pas modifier directement le type de données d'une colonne. À la place, vous recréez la table Athena et définissez la colonne avec le nouveau type de données.
- Seuls certains types de données peuvent être lus dans d'autres types de données. Consultez la table de cette section pour les types de données qui peuvent traités de cette manière.
- Pour les données au format Parquet et ORC, vous ne pouvez pas utiliser un autre type de données pour une colonne si la table n'est pas partitionnée.
- Pour les tables partitionnées dans Parquet et ORC, le type de colonne d'une partition peut être différent de celui d'une autre partition, et Athena convertira (`CAST`) au type souhaité, si possible.

Pour plus d’informations, veuillez consulter Éviter les erreurs de non-correspondance de schéma pour les tables avec des partitions.

- Pour les tables créées à l'aide de l'instruction LazySimpleSerDeonly, il est possible d'utiliser l'`ALTER TABLE REPLACE COLUMNS`instruction pour remplacer les colonnes existantes par un type de données différent, mais toutes les colonnes existantes que vous souhaitez conserver doivent également être redéfinies dans l'instruction, sinon elles seront supprimées. Pour plus d’informations, consultez ALTER TABLE REPLACE COLUMNS.
- Pour les tables Apache Iceberg uniquement, vous pouvez utiliser l'instruction ALTER TABLE CHANGE COLUMNpour modifier le type de données d'une colonne. `ALTER TABLE REPLACE COLUMNS` n'est pas pris en charge pour les tables Iceberg. Pour plus d’informations, consultez Schéma évolutif de la table Iceberg.

> **Important**
>
> Nous vous recommandons vivement de tester et de vérifier vos requêtes avant d'effectuer des conversions de type de données. Si Athena ne peut pas utiliser le type de données cible, la requête `CREATE TABLE` risque d'échouer.

Le tableau suivant répertorie les types de données qui doivent être traités comme autres types de données :

Types de données compatibles

Type de données d'origine	Types de données cibles disponibles
`STRING`	`BYTE`, `TINYINT`, `SMALLINT`, `INT`, `BIGINT`
`BYTE`	`TINYINT`, `SMALLINT`, `INT`, `BIGINT`
`TINYINT`	`SMALLINT`, `INT`, `BIGINT`
`SMALLINT`	`INT`, `BIGINT`
`INT`	`BIGINT`
`FLOAT`	`DOUBLE`

L'exemple suivant utilise l'instruction `CREATE TABLE` de la table `orders_json` d'origine pour créer une nouvelle table appelée `orders_json_bigint`. La nouvelle table utilise `BIGINT` plutôt que `INT` comme type de données pour la colonne `` `o_shippriority` ``.

```
CREATE EXTERNAL TABLE orders_json_bigint (
   `o_orderkey` int,
   `o_custkey` int,
   `o_orderstatus` string,
   `o_totalprice` double,
   `o_orderdate` string,
   `o_orderpriority` string,
   `o_clerk` string,
   `o_shippriority` BIGINT
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_json';
```

La requête suivante s'exécute correctement, comme dans la requête `SELECT` d'origine, avant le changement du type de données :

```
Select * from orders_json
LIMIT 10;
```

Mises à jour dans les tables avec des partitions

Dans Athena, une table et ses partitions doivent utiliser les mêmes formats de données, mais leurs schémas peuvent différer. Lorsque vous créez une nouvelle partition, cette partition hérite généralement du schéma de la table. Au fil du temps, les schémas peuvent commencer à différer. Les raisons sont les suivantes :

- Si le schéma votre table change, les schémas des partitions ne sont pas mis à jour afin de rester synchronisés avec le schéma de la table.
- Le AWS Glue Crawler vous permet de découvrir des données dans des partitions dotées de différents schémas. Cela signifie que si vous créez une table dans Athena avec AWS Glue, une fois le traitement terminé par le robot, les schémas de la table et de ses partitions peuvent être différents.
- Si vous ajoutez des partitions directement à l'aide d'une AWS API.

Athena traite les tables avec des partitions correctement si elles répondent aux contraintes suivantes. Si ces contraintes ne sont pas satisfaites, Athena émet une erreur HIVE_PARTITION_SCHEMA_MISMATCH.

- Le schéma de chaque partition est compatible avec le schéma de la table.
- Le format des données de la table autorise le type de mise à jour que vous souhaitez réaliser : ajouter, supprimer, réorganiser les colonnes, ou modifier le type de données d'une colonne.

  Par exemple, pour les formats CSV et TSV, vous pouvez uniquement renommer les colonnes, ajouter de nouvelles colonnes à la fin de la table, et modifier le type de données d'une colonne si les types sont compatibles, mais vous ne pouvez pas supprimer des colonnes. Pour les autres formats, vous pouvez ajouter ou supprimer des colonnes, ou remplacer le type de données d'une colonne par un autre si les types sont compatibles. Pour plus d'informations, consultez la rubrique Récapitulatif : mises à jour et formats de données dans Athena.

Éviter les erreurs de non-correspondance de schéma pour les tables avec des partitions

Au début de l'exécution des requêtes, Athena vérifie le schéma de la table en s'assurant que le type de données de chaque colonne est compatible entre la table et la partition.

- Pour les types de stockage de données Parquet et ORC, Athena se base sur les noms de colonne et les utilise pour la vérification du schéma basée sur les noms de colonne. Cela élimine les erreurs `HIVE_PARTITION_SCHEMA_MISMATCH` pour les tables avec des partitions de type Parquet ou ORC. (Cela est vrai pour ORC si la SerDe propriété est définie pour accéder à l'index par son nom :`orc.column.index.access=FALSE`. Parquet lit l'index par nom (par défaut).
- Pour CSV, JSON et Avro, Athena utilise une vérification du schéma basée sur un index. Cela signifie que si vous rencontrez une erreur de non-correspondance de schéma, vous devez supprimer la partition qui provoque une non-correspondance de schéma et la recréer, de telle sorte qu'Athena puisse l'interroger sans risque d'échec.

Athena compare le schéma de la table aux schémas de la partition. Si vous créez une table au format CSV, JSON et AVRO dans Athena AWS Glue avec Crawler, une fois le traitement terminé, les schémas de la table et de ses partitions peuvent être différents. En cas de non-correspondance entre le schéma de la table et les schémas de partition, les requêtes échouent dans Athena en raison d'une erreur de vérification de schéma semblable à celle-ci : 'crawler_test.click_avro' is declared as type 'string', but partition 'partition_0=2017-01-17' declared column 'col68'

as type 'double' (crawler_test.click_avro' est déclaré comme type 'chaîne', mais la partition 'partition_0=2017-01-17' a déclaré la colonne 'col68' comme type 'double').

Une solution de contournement pour ce type d'erreurs consiste à supprimer la partition qui provoque l'erreur et à la recréer. Pour plus d'informations, consultez ALTER TABLE DROP PARTITION et ALTER TABLE ADD PARTITION.

Interrogation des tableaux

Amazon Athena vous permet de créer des tableaux, de les concaténer, de les convertir en différents types de données, puis de les filtrer, les aplatir et les trier.

Rubriques

Création de tableaux

Pour créer un libellé tableau dans Athena, utilisez le mot clé `ARRAY`, suivi de crochets `[ ]`, et incluez les éléments du tableau séparés par des virgules.

Exemples

Cette requête crée un tableau avec quatre éléments.

```
SELECT ARRAY [1,2,3,4] AS items
```

Elle renvoie :

```
+-----------+
| items     |
+-----------+
| [1,2,3,4] |
+-----------+
```

Cette requête crée deux tableaux.

```
SELECT ARRAY[ ARRAY[1,2], ARRAY[3,4] ] AS items
```

Elle renvoie :

```
+--------------------+
| items              |
+--------------------+
| [[1, 2], [3, 4]]   |
+--------------------+
```

Pour créer un tableau à partir de colonnes sélectionnées de types compatibles, utilisez une requête, comme dans cet exemple :

```
WITH
dataset AS (
  SELECT 1 AS x, 2 AS y, 3 AS z
)
SELECT ARRAY [x,y,z] AS items FROM dataset
```

Cette requête renvoie :

```
+-----------+
| items     |
+-----------+
| [1,2,3]   |
+-----------+
```

Dans l'exemple suivant, deux tableaux sont sélectionnés et retournés comme message de bienvenue.

```
WITH
dataset AS (
  SELECT
    ARRAY ['hello', 'amazon', 'athena'] AS words,
    ARRAY ['hi', 'alexa'] AS alexa
)
SELECT ARRAY[words, alexa] AS welcome_msg
FROM dataset
```

Cette requête renvoie :

```
+----------------------------------------+
| welcome_msg                            |
+----------------------------------------+
| [[hello, amazon, athena], [hi, alexa]] |
+----------------------------------------+
```

Pour créer un tableau de paires clé-valeur, utilisez l'opérateur MAP qui accepte un tableau de clés, suivi d'un tableau de valeurs, comme dans cet exemple :

```
SELECT ARRAY[
   MAP(ARRAY['first', 'last', 'age'],ARRAY['Bob', 'Smith', '40']),
   MAP(ARRAY['first', 'last', 'age'],ARRAY['Jane', 'Doe', '30']),
   MAP(ARRAY['first', 'last', 'age'],ARRAY['Billy', 'Smith', '8'])
] AS people
```

Cette requête renvoie :

```
+-------------------------------------------------------------------------------------------------------------
+
| people
               |
+-------------------------------------------------------------------------------------------------------------
+
| [{last=Smith, first=Bob, age=40}, {last=Doe, first=Jane, age=30}, {last=Smith,
 first=Billy, age=8}] |
+-------------------------------------------------------------------------------------------------------------
+
```

Concaténation de chaînes et de tableaux

Concaténation de chaînes

Pour concaténer deux chaînes, vous pouvez utiliser l'opérateur à deux barres verticales `||` comme dans l'exemple suivant.

```
SELECT 'This' || ' is' || ' a' || ' test.' AS Concatenated_String
```

Cette requête renvoie :

#	Concatenated_String
1	This is a test.

Vous pouvez utiliser la fonction `concat()` pour obtenir le même résultat.

```
SELECT concat('This', ' is', ' a', ' test.') AS Concatenated_String
```

Cette requête renvoie :

#	Concatenated_String
1	This is a test.

Vous pouvez utiliser la fonction `concat_ws()` pour concaténer des chaînes avec le séparateur spécifié dans le premier argument.

```
SELECT concat_ws(' ', 'This', 'is', 'a', 'test.') as Concatenated_String
```

Cette requête renvoie :

#	Concatenated_String
1	This is a test.

Pour concaténer deux colonnes du type de données chaîne à l'aide d'un point, référencez les deux colonnes à l'aide de guillemets doubles et placez le point entre guillemets simples en tant que chaîne codée en dur. Si une colonne n'est pas du type de données chaîne, vous pouvez utiliser `CAST("column_name" as VARCHAR)` pour convertir la colonne au préalable.

```
SELECT "col1" || '.' || "col2" as Concatenated_String
FROM my_table
```

Cette requête renvoie :

#	Concatenated_String
1	*col1_string_value* .*col2_string_value*

Concaténation de tableaux

Vous pouvez utiliser les mêmes techniques pour concaténer des tableaux.

Pour concaténer plusieurs tableaux, utilisez l'opérateur à deux barres verticales `||`.

```
SELECT ARRAY [4,5] || ARRAY[ ARRAY[1,2], ARRAY[3,4] ] AS items
```

Cette requête renvoie :

#	items
1	[[4, 5], [1, 2], [3, 4]]

Pour combiner plusieurs tableaux en un seul tableau, utilisez l'opérateur à deux barres ou la fonction `concat()`.

```
WITH
dataset AS (
  SELECT
    ARRAY ['Hello', 'Amazon', 'Athena'] AS words,
    ARRAY ['Hi', 'Alexa'] AS alexa
```

```
)
SELECT concat(words, alexa) AS welcome_msg
FROM dataset
```

Cette requête renvoie :

#	welcome_msg
1	[Hello, Amazon, Athena, Hi, Alexa]

Pour plus d'informations sur les autres fonctions `concat()` de chaîne, veuillez consulter la rubrique String functions and operators dans la documentation de Trino.

Conversion des types de données de tableau

Pour convertir des données de tableaux en types de données pris en charge, utilisez l'opérateur `CAST`, par exemple `CAST(value AS type)`. Athena prend en charge tous les types de données Presto natifs.

```
SELECT
   ARRAY [CAST(4 AS VARCHAR), CAST(5 AS VARCHAR)]
AS items
```

Cette requête renvoie :

```
+-------+
| items |
+-------+
| [4,5] |
+-------+
```

Créez deux tableaux avec des éléments de paire clé-valeur, convertissez-les en JSON et concaténez-les, comme dans cet exemple :

```
SELECT
   ARRAY[CAST(MAP(ARRAY['a1', 'a2', 'a3'], ARRAY[1, 2, 3]) AS JSON)] ||
   ARRAY[CAST(MAP(ARRAY['b1', 'b2', 'b3'], ARRAY[4, 5, 6]) AS JSON)]
AS items
```

Cette requête renvoie :

```
+-------------------------------------------------+
| items                                           |
+-------------------------------------------------+
| [{"a1":1,"a2":2,"a3":3}, {"b1":4,"b2":5,"b3":6}] |
+-------------------------------------------------+
```

Recherche des longueurs

La fonction `cardinality` renvoie la longueur d'un tableau, comme dans cet exemple :

```
SELECT cardinality(ARRAY[1,2,3,4]) AS item_count
```

Cette requête renvoie :

```
+------------+
| item_count |
+------------+
| 4          |
+------------+
```

Accès à des éléments de tableau

Pour accéder à des éléments de tableau, utilisez l'opérateur `[]`, avec 1 spécifiant le premier élément, 2 le deuxième élément, et ainsi de suite, comme dans cet exemple :

```
WITH dataset AS (
SELECT
   ARRAY[CAST(MAP(ARRAY['a1', 'a2', 'a3'], ARRAY[1, 2, 3]) AS JSON)] ||
   ARRAY[CAST(MAP(ARRAY['b1', 'b2', 'b3'], ARRAY[4, 5, 6]) AS JSON)]
AS items )
SELECT items[1] AS item FROM dataset
```

Cette requête renvoie :

```
+------------------------+
| item                   |
+------------------------+
| {"a1":1,"a2":2,"a3":3} |
```

```
+------------------------+
```

Pour accéder aux éléments d'un tableau à une position donnée (appelée position d'index), utilisez la fonction `element_at()`, puis spécifiez le nom du tableau et la position d'index :

- Si l'index est supérieur à 0, `element_at()` renvoie l'élément que vous spécifiez, en comptant du début à la fin du tableau. La fonction se comporte comme l'opérateur `[]`.
- Si l'index est inférieur à 0, `element_at()` renvoie l'élément que vous spécifiez, en comptant de la fin au début du tableau.

La requête suivante crée un tableau `words`, puis sélectionne dans celui-ci le premier élément `hello` en tant que `first_word`, le deuxième élément `amazon` (en partant de la fin du tableau) en tant que `middle_word` et le troisième élément `athena` en tant que `last_word`.

```
WITH dataset AS (
  SELECT ARRAY ['hello', 'amazon', 'athena'] AS words
)
SELECT
  element_at(words, 1) AS first_word,
  element_at(words, -2) AS middle_word,
  element_at(words, cardinality(words)) AS last_word
FROM dataset
```

Cette requête renvoie :

```
+----------------------------------------+
| first_word  | middle_word | last_word  |
+----------------------------------------+
| hello       | amazon      | athena     |
+----------------------------------------+
```

Aplatissement de tableaux imbriqués

Lorsque vous utilisez des tableaux imbriqués, vous avez souvent besoin de développer des éléments de tableau imbriqué dans un seul tableau, ou de développer le tableau en plusieurs lignes.

Exemples

Pour aplatir les éléments d'un tableau imbriqué dans un seul tableau de valeurs, utilisez la fonction `flatten`. Cette requête renvoie une ligne pour chaque élément du tableau.

```
SELECT flatten(ARRAY[ ARRAY[1,2], ARRAY[3,4] ]) AS items
```

Cette requête renvoie :

```
+-----------+
| items     |
+-----------+
| [1,2,3,4] |
+-----------+
```

Pour aplatir un tableau en plusieurs lignes, utilisez `CROSS JOIN` conjointement avec l'opérateur `UNNEST`, comme dans cet exemple :

```
WITH dataset AS (
  SELECT
    'engineering' as department,
    ARRAY['Sharon', 'John', 'Bob', 'Sally'] as users
)
SELECT department, names FROM dataset
CROSS JOIN UNNEST(users) as t(names)
```

Cette requête renvoie :

```
+----------------------+
| department  | names  |
+----------------------+
| engineering | Sharon |
+----------------------|
| engineering | John   |
+----------------------|
| engineering | Bob    |
+----------------------|
| engineering | Sally  |
+----------------------+
```

Pour aplatir un tableau de paires clé-valeur, transposez les clés sélectionnées en colonnes, comme dans cet exemple :

```
WITH
dataset AS (
```

```
  SELECT
    'engineering' as department,
     ARRAY[
      MAP(ARRAY['first', 'last', 'age'],ARRAY['Bob', 'Smith', '40']),
      MAP(ARRAY['first', 'last', 'age'],ARRAY['Jane', 'Doe', '30']),
      MAP(ARRAY['first', 'last', 'age'],ARRAY['Billy', 'Smith', '8'])
     ] AS people
  )
SELECT names['first'] AS
 first_name,
 names['last'] AS last_name,
 department FROM dataset
CROSS JOIN UNNEST(people) AS t(names)
```

Cette requête renvoie :

```
+--------------------------------------+
| first_name | last_name | department  |
+--------------------------------------+
| Bob        | Smith     | engineering |
| Jane       | Doe       | engineering |
| Billy      | Smith     | engineering |
+--------------------------------------+
```

Depuis la liste des employés, sélectionnez l'employé ayant obtenu les meilleurs scores combinés. UNNEST peut être utilisé dans la clause FROM sans être précédé par CROSS JOIN, car il s'agit de l'opérateur de jointure par défaut qui est donc implicite.

```
WITH
dataset AS (
  SELECT ARRAY[
    CAST(ROW('Sally', 'engineering', ARRAY[1,2,3,4]) AS ROW(name VARCHAR, department
 VARCHAR, scores ARRAY(INTEGER))),
    CAST(ROW('John', 'finance', ARRAY[7,8,9]) AS ROW(name VARCHAR, department VARCHAR,
 scores ARRAY(INTEGER))),
    CAST(ROW('Amy', 'devops', ARRAY[12,13,14,15]) AS ROW(name VARCHAR, department
 VARCHAR, scores ARRAY(INTEGER)))
  ] AS users
),
users AS (
 SELECT person, score
 FROM
```

```
   dataset,
   UNNEST(dataset.users) AS t(person),
   UNNEST(person.scores) AS t(score)
)
SELECT person.name, person.department, SUM(score) AS total_score FROM users
GROUP BY (person.name, person.department)
ORDER BY (total_score) DESC
LIMIT 1
```

Cette requête renvoie :

```
+---------------------------------+
| name | department | total_score |
+---------------------------------+
| Amy  | devops     | 54          |
+---------------------------------+
```

Depuis la liste des employés, sélectionnez l'employé ayant obtenu le meilleur score individuel.

```
WITH
dataset AS (
 SELECT ARRAY[
   CAST(ROW('Sally', 'engineering', ARRAY[1,2,3,4]) AS ROW(name VARCHAR, department
 VARCHAR, scores ARRAY(INTEGER))),
   CAST(ROW('John', 'finance', ARRAY[7,8,9]) AS ROW(name VARCHAR, department VARCHAR,
 scores ARRAY(INTEGER))),
   CAST(ROW('Amy', 'devops', ARRAY[12,13,14,15]) AS ROW(name VARCHAR, department
 VARCHAR, scores ARRAY(INTEGER)))
 ] AS users
),
users AS (
 SELECT person, score
 FROM
   dataset,
   UNNEST(dataset.users) AS t(person),
   UNNEST(person.scores) AS t(score)
)
SELECT person.name, score FROM users
ORDER BY (score) DESC
LIMIT 1
```

Cette requête renvoie :

```
+--------------+
| name | score |
+--------------+
| Amy  | 15    |
+--------------+
```

Considérations et restrictions

Si UNNEST est utilisé sur un ou plusieurs tableaux de la requête, et que l'un des tableaux est NULL, la requête ne renvoie aucune ligne. Si UNNEST est utilisé sur un tableau qui est une chaîne vide, la chaîne vide est renvoyée.

Par exemple, dans la requête suivante, le deuxième tableau étant null, la requête ne renvoie aucune ligne.

```
SELECT
    col1,
    col2
FROM UNNEST (ARRAY ['apples','oranges','lemons']) AS t(col1)
CROSS JOIN UNNEST (ARRAY []) AS t(col2)
```

Dans l'exemple suivant, le deuxième tableau est modifié pour contenir une chaîne vide. Pour chaque ligne, la requête renvoie la valeur dans `col1` et une chaîne vide pour la valeur dans `col2`. La chaîne vide du deuxième tableau est requise pour que les valeurs du premier tableau soient renvoyées.

```
SELECT
    col1,
    col2
FROM UNNEST (ARRAY ['apples','oranges','lemons']) AS t(col1)
CROSS JOIN UNNEST (ARRAY ['']) AS t(col2)
```

Création de tableaux à partir de sous-requêtes

Créez un tableau à partir d'un ensemble de lignes.

```
WITH
dataset AS (
  SELECT ARRAY[1,2,3,4,5] AS items
)
SELECT array_agg(i) AS array_items
```

```
FROM dataset
CROSS JOIN UNNEST(items) AS t(i)
```

Cette requête renvoie :

```
+-----------------+
| array_items     |
+-----------------+
| [1, 2, 3, 4, 5] |
+-----------------+
```

Pour créer un tableau de valeurs uniques à partir d'un ensemble de lignes, utilisez le mot-clé `distinct`.

```
WITH
dataset AS (
  SELECT ARRAY [1,2,2,3,3,4,5] AS items
)
SELECT array_agg(distinct i) AS array_items
FROM dataset
CROSS JOIN UNNEST(items) AS t(i)
```

Cette requête renvoie le résultat suivant : Notez que l'ordre n'est pas garanti.

```
+-----------------+
| array_items     |
+-----------------+
| [1, 2, 3, 4, 5] |
+-----------------+
```

Pour plus d'informations sur l'utilisation de la fonction `array_agg`, veuillez consulter Fonctions d'agrégation dans la documentation de Trino.

Filtrage des tableaux

Créez un tableau à partir d'un ensemble de lignes si elles correspondent aux critères de filtre.

```
WITH
dataset AS (
  SELECT ARRAY[1,2,3,4,5] AS items
)
```

```
SELECT array_agg(i) AS array_items
FROM dataset
CROSS JOIN UNNEST(items) AS t(i)
WHERE i > 3
```

Cette requête renvoie :

```
+-------------+
| array_items |
+-------------+
| [4, 5]      |
+-------------+
```

Filtrez un tableau selon que l'un de ses éléments contient une valeur spécifique, telle que 2, comme dans cet exemple :

```
WITH
dataset AS (
  SELECT ARRAY
  [
    ARRAY[1,2,3,4],
    ARRAY[5,6,7,8],
    ARRAY[9,0]
  ] AS items
)
SELECT i AS array_items FROM dataset
CROSS JOIN UNNEST(items) AS t(i)
WHERE contains(i, 2)
```

Cette requête renvoie :

```
+--------------+
| array_items  |
+--------------+
| [1, 2, 3, 4] |
+--------------+
```

La fonction `filter`

```
 filter(ARRAY [list_of_values], boolean_function)
```

Vous pouvez utiliser la fonction `filter` sur une expression ARRAY pour créer une nouvelle table qui est le sous-ensemble des éléments du tableau de la liste *`list_of_values`* pour lequel la fonction *`boolean_function`* est vraie. La fonction `filter` peut être utile dans les cas où vous ne pouvez pas utiliser la fonction *`UNNEST`*.

L'exemple suivant filtre les valeurs supérieures à zéro dans le tableau `[1,0,5,-1]`.

```
SELECT filter(ARRAY [1,0,5,-1], x -> x>0)
```

Résultats

```
[1,5]
```

L'exemple suivant filtre les valeurs non nulles dans le tableau `[-1, NULL, 10, NULL]`.

```
SELECT filter(ARRAY [-1, NULL, 10, NULL], q -> q IS NOT NULL)
```

Résultats

```
[-1,10]
```

Tri de tableaux

Pour créer un tableau trié de valeurs uniques à partir d'un ensemble de lignes, vous pouvez utiliser la fonction array_sort, comme dans l'exemple suivant.

```
WITH
dataset AS (
  SELECT ARRAY[3,1,2,5,2,3,6,3,4,5] AS items
)
SELECT array_sort(array_agg(distinct i)) AS array_items
FROM dataset
CROSS JOIN UNNEST(items) AS t(i)
```

Cette requête renvoie :

```
+--------------------+
| array_items        |
+--------------------+
| [1, 2, 3, 4, 5, 6] |
```

```
+--------------------+
```

Pour plus d'informations sur l'extension d'un tableau en plusieurs lignes, consultez Aplatissement de tableaux imbriqués.

Utilisation des fonctions d'agrégation avec des tableaux

- Pour ajouter des valeurs figurant dans un tableau, utilisez SUM, comme dans l'exemple suivant.
- Pour regrouper plusieurs lignes dans un tableau, utilisez `array_agg`. Pour plus d'informations, consultez Création de tableaux à partir de sous-requêtes.

> **Note**
>
> La commande `ORDER BY` est prise en charge pour les fonctions d'agrégation à partir de la version 2 du moteur Athena.

```
WITH
dataset AS (
  SELECT ARRAY
  [
    ARRAY[1,2,3,4],
    ARRAY[5,6,7,8],
    ARRAY[9,0]
  ] AS items
),
item AS (
  SELECT i AS array_items
  FROM dataset, UNNEST(items) AS t(i)
)
SELECT array_items, sum(val) AS total
FROM item, UNNEST(array_items) AS t(val)
GROUP BY array_items;
```

Dans la dernière instruction SELECT, au lieu d'utiliser `sum()` et UNNEST, vous pouvez utiliser `reduce()` pour réduire le temps de traitement et le transfert de données, comme dans l'exemple suivant.

```
WITH
```

```
dataset AS (
  SELECT ARRAY
  [
    ARRAY[1,2,3,4],
    ARRAY[5,6,7,8],
    ARRAY[9,0]
  ] AS items
),
item AS (
  SELECT i AS array_items
  FROM dataset, UNNEST(items) AS t(i)
)
SELECT array_items, reduce(array_items, 0 , (s, x) -> s + x, s -> s) AS total
FROM item;
```

La requête renvoie les résultats suivants. L'ordre des résultats renvoyés n'est pas garanti.

```
+----------------------+
| array_items  | total |
+----------------------+
| [1, 2, 3, 4] | 10    |
| [5, 6, 7, 8] | 26    |
| [9, 0]       | 9     |
+----------------------+
```

Conversion de tableaux en chaînes

Pour convertir un tableau en une seule chaîne, utilisez la fonction `array_join`. L'exemple autonome suivant crée une table appelée `dataset` qui contient un tableau alias appelé `words`. La requête utilise `array_join` pour joindre les éléments du tableau dans `words`, les séparer par des espaces et renvoyer la chaîne résultante dans une colonne alias appelée `welcome_msg`.

```
WITH
dataset AS (
  SELECT ARRAY ['hello', 'amazon', 'athena'] AS words
)
SELECT array_join(words, ' ') AS welcome_msg
FROM dataset
```

Cette requête renvoie :

```
+---------------------+
```

```
| welcome_msg         |
+---------------------+
| hello amazon athena |
+---------------------+
```

Utilisation de tableaux pour créer des mappages

Les mappages sont des paires valeur-clé constituées de types de données disponibles dans Athena. Pour créer des mappages, utilisez l'opérateur `MAP` et transmettez-lui deux tableaux : le premier tableau comprend des noms de colonne (clé) et le deuxième comprend des valeurs. Toutes les valeurs des tableaux doivent être du même type. Si des éléments d'un tableau de mappage de valeurs doivent être de types différents, vous pourrez les convertir ultérieurement.

Exemples

Cet exemple sélectionne un utilisateur à partir d'un ensemble de données. Il utilise l'opérateur `MAP` et lui transmet deux tableaux. Le premier tableau comprend des valeurs pour des noms de colonne, comme « first », « last » et « age ». Le deuxième tableau comprend des valeurs pour chacune de ces colonnes, comme « Bob », « Smith » et « 35 ».

```
WITH dataset AS (
  SELECT MAP(
    ARRAY['first', 'last', 'age'],
    ARRAY['Bob', 'Smith', '35']
  ) AS user
)
SELECT user FROM dataset
```

Cette requête renvoie :

```
+--------------------------------+
| user                           |
+--------------------------------+
| {last=Smith, first=Bob, age=35} |
+--------------------------------+
```

Vous pouvez récupérer les valeurs de `Map` en sélectionnant le nom du champ suivi par `[key_name]`, comme dans cet exemple :

```
WITH dataset AS (
```

```
 SELECT MAP(
   ARRAY['first', 'last', 'age'],
   ARRAY['Bob', 'Smith', '35']
 ) AS user
)
SELECT user['first'] AS first_name FROM dataset
```

Cette requête renvoie :

```
+------------+
| first_name |
+------------+
| Bob        |
+------------+
```

Interrogation de tableaux avec des types complexes et des structures imbriquées

Vos données sources contiennent souvent des tableaux avec des types de données complexes et des structures imbriquées. Les exemples de cette section montrent comment modifier le type de données de l'élément, rechercher des éléments au sein de tableaux et trouver des mots-clés à l'aide de requêtes Athena.

- Création d'une `ROW`
- Modification des noms de champ de tableaux avec `CAST`
- Filtrage des tableaux à l'aide de la notation `.`
- Filtrage des tableaux avec les valeurs imbriquées
- Filtrage des tableaux avec `UNNEST`
- Recherche de mots clés dans les tableaux avec `regexp_like`

Création d'une **`ROW`**

> **Note**
>
> Les exemples de cette section utilisent `ROW` pour créer des exemples de données à utiliser. Lorsque vous interrogez les tables dans Athena, vous n'avez pas besoin de créer des types de données `ROW`, car ils sont déjà créés à partir de votre source de données. Lorsque vous utilisez la commande `CREATE_TABLE`, Athena y définit un `STRUCT`, renseigne les données et

crée automatiquement le type de données `ROW` pour chaque ligne du jeu de données. Le type de données `ROW` sous-jacent se compose de champs nommés de tous les types de données SQL pris en charge.

```
WITH dataset AS (
 SELECT
   ROW('Bob', 38) AS users
 )
SELECT * FROM dataset
```

Cette requête renvoie :

```
+-------------------------+
| users                   |
+-------------------------+
| {field0=Bob, field1=38} |
+-------------------------+
```

Modification des noms de champ de tableaux avec **`CAST`**

Pour modifier le nom de champ d'un tableau qui contient les valeurs `ROW`, vous pouvez `CAST` la déclaration `ROW` :

```
WITH dataset AS (
  SELECT
    CAST(
      ROW('Bob', 38) AS ROW(name VARCHAR, age INTEGER)
    ) AS users
)
SELECT * FROM dataset
```

Cette requête renvoie :

```
+--------------------+
| users              |
+--------------------+
| {NAME=Bob, AGE=38} |
+--------------------+
```

> **Note**
>
> Dans l'exemple ci-dessus, vous déclarez `name` en tant que `VARCHAR` parce que c'est son type dans Presto. Si vous déclarez ce `STRUCT` à l'intérieur d'une instruction `CREATE TABLE`, utilisez le type `String` parce que Hive définit ce type de données comme `String`.

Filtrage des tableaux à l'aide de la notation `.`

Dans l'exemple suivant, sélectionnez le champ `accountId` de la colonne `userIdentity` d'une table de journaux AWS CloudTrail à l'aide de la notation `.` (point). Pour de plus amples informations, consultez la rubrique Interrogation des journaux AWS CloudTrail.

```
SELECT
  CAST(useridentity.accountid AS bigint) as newid
FROM cloudtrail_logs
LIMIT 2;
```

Cette requête renvoie :

```
+--------------+
| newid        |
+--------------+
| 112233445566 |
+--------------+
| 998877665544 |
+--------------+
```

Pour interroger un tableau de valeurs, exécutez cette requête :

```
WITH dataset AS (
  SELECT ARRAY[
    CAST(ROW('Bob', 38) AS ROW(name VARCHAR, age INTEGER)),
    CAST(ROW('Alice', 35) AS ROW(name VARCHAR, age INTEGER)),
    CAST(ROW('Jane', 27) AS ROW(name VARCHAR, age INTEGER))
  ] AS users
)
SELECT * FROM dataset
```

Elle renvoie le résultat suivant :

```
+-------------------------------------------------------------------+
| users                                                             |
+-------------------------------------------------------------------+
| [{NAME=Bob, AGE=38}, {NAME=Alice, AGE=35}, {NAME=Jane, AGE=27}] |
+-------------------------------------------------------------------+
```

Filtrage des tableaux avec les valeurs imbriquées

Les grands tableaux contiennent souvent des structures imbriquées et vous devez pouvoir y filtrer ou y rechercher des valeurs.

Pour définir un ensemble de données pour un tableau de valeurs qui inclut une valeur BOOLEAN imbriquée, exécutez cette requête :

```
WITH dataset AS (
  SELECT
    CAST(
      ROW('aws.amazon.com', ROW(true)) AS ROW(hostname VARCHAR, flaggedActivity
 ROW(isNew BOOLEAN))
    ) AS sites
)
SELECT * FROM dataset
```

Elle renvoie le résultat suivant :

```
+-----------------------------------------------------------+
| sites                                                     |
+-----------------------------------------------------------+
| {HOSTNAME=aws.amazon.com, FLAGGEDACTIVITY={ISNEW=true}}  |
+-----------------------------------------------------------+
```

Ensuite, pour filtrer la valeur BOOLEAN de cet élément et y accéder, continuez à utiliser la notation . (point).

```
WITH dataset AS (
  SELECT
    CAST(
      ROW('aws.amazon.com', ROW(true)) AS ROW(hostname VARCHAR, flaggedActivity
 ROW(isNew BOOLEAN))
    ) AS sites
```

```
)
SELECT sites.hostname, sites.flaggedactivity.isnew
FROM dataset
```

Cette requête sélectionne les champs imbriqués et renvoie les résultats suivants :

```
+------------------------+
| hostname       | isnew |
+------------------------+
| aws.amazon.com | true  |
+------------------------+
```

Filtrage des tableaux avec **UNNEST**

Pour filtrer un tableau qui inclut une structure imbriquée par l'un de ses éléments enfants, émettez une requête avec un opérateur UNNEST. Pour en savoir plus sur UNNEST, consultez Aplatissement de tableaux imbriqués.

Par exemple, cette requête retrouve les noms d'hôte des sites dans le jeu de données.

```
WITH dataset AS (
  SELECT ARRAY[
    CAST(
      ROW('aws.amazon.com', ROW(true)) AS ROW(hostname VARCHAR, flaggedActivity
 ROW(isNew BOOLEAN))
    ),
    CAST(
      ROW('news.cnn.com', ROW(false)) AS ROW(hostname VARCHAR, flaggedActivity
 ROW(isNew BOOLEAN))
    ),
    CAST(
      ROW('netflix.com', ROW(false)) AS ROW(hostname VARCHAR, flaggedActivity ROW(isNew
 BOOLEAN))
    )
  ] as items
)
SELECT sites.hostname, sites.flaggedActivity.isNew
FROM dataset, UNNEST(items) t(sites)
WHERE sites.flaggedActivity.isNew = true
```

Elle renvoie :

```
+------------------------+
| hostname       | isnew |
+------------------------+
| aws.amazon.com | true  |
+------------------------+
```

Recherche de mots clés dans les tableaux avec **`regexp_like`**

Les exemples suivants illustrent comment explorer un jeu de données à la recherche d'un mot-clé dans un élément à l'intérieur d'un tableau, à l'aide de la fonction regexp_like. Il prend comme entrée le modèle d'une expression régulière à évaluer, ou une liste de termes séparés par une barre (|), évalue le modèle, et détermine si la chaîne spécifiée le contient.

Le modèle d'expression régulière doit être contenu dans la chaîne, mais n'a pas besoin de lui correspondre. Pour faire correspondre l'ensemble de la chaîne, placez le modèle entre les symboles ^ et $, par exemple `'^pattern$'`.

Considérons une matrice des sites contenant leur nom d'hôte et un élément `flaggedActivity`. Cet élément comprend un ARRAY, contenant plusieurs éléments MAP, chacun offrant différents mots-clés populaires et leur nombre de popularité. Supposons que vous vouliez trouver un mot-clé particulier à l'intérieur d'une MAP de ce tableau.

Pour chercher des sites dans cet ensemble de données, avec un mot-clé particulier, nous utilisons `regexp_like` au lieu de l'opérateur SQL similaire `LIKE`, car la recherche avec plusieurs mots-clés est plus efficace avec `regexp_like`.

Example Exemple 1 : à l'aide de **`regexp_like`**

La requête de cet exemple utilise la fonction `regexp_like` pour rechercher des termes `'politics|bigdata'`, détectés dans les valeurs au sein des tableaux :

```
WITH dataset AS (
  SELECT ARRAY[
    CAST(
      ROW('aws.amazon.com', ROW(ARRAY[
          MAP(ARRAY['term', 'count'], ARRAY['bigdata', '10']),
          MAP(ARRAY['term', 'count'], ARRAY['serverless', '50']),
          MAP(ARRAY['term', 'count'], ARRAY['analytics', '82']),
          MAP(ARRAY['term', 'count'], ARRAY['iot', '74'])
      ])
```

```
      ) AS ROW(hostname VARCHAR, flaggedActivity ROW(flags ARRAY(MAP(VARCHAR,
 VARCHAR)) ))
   ),
   CAST(
     ROW('news.cnn.com', ROW(ARRAY[
       MAP(ARRAY['term', 'count'], ARRAY['politics', '241']),
       MAP(ARRAY['term', 'count'], ARRAY['technology', '211']),
       MAP(ARRAY['term', 'count'], ARRAY['serverless', '25']),
       MAP(ARRAY['term', 'count'], ARRAY['iot', '170'])
     ])
     ) AS ROW(hostname VARCHAR, flaggedActivity ROW(flags ARRAY(MAP(VARCHAR,
 VARCHAR)) ))
   ),
   CAST(
     ROW('netflix.com', ROW(ARRAY[
       MAP(ARRAY['term', 'count'], ARRAY['cartoons', '1020']),
       MAP(ARRAY['term', 'count'], ARRAY['house of cards', '112042']),
       MAP(ARRAY['term', 'count'], ARRAY['orange is the new black', '342']),
       MAP(ARRAY['term', 'count'], ARRAY['iot', '4'])
     ])
     ) AS ROW(hostname VARCHAR, flaggedActivity ROW(flags ARRAY(MAP(VARCHAR,
 VARCHAR)) ))
   )
 ] AS items
),
sites AS (
  SELECT sites.hostname, sites.flaggedactivity
  FROM dataset, UNNEST(items) t(sites)
)
SELECT hostname
FROM sites, UNNEST(sites.flaggedActivity.flags) t(flags)
WHERE regexp_like(flags['term'], 'politics|bigdata')
GROUP BY (hostname)
```

Cette requête renvoie deux sites :

```
+----------------+
| hostname       |
+----------------+
| aws.amazon.com |
+----------------+
| news.cnn.com   |
+----------------+
```

Example Exemple 2 : à l'aide de **`regexp_like`**

La requête de l'exemple suivant ajoute le total des scores de popularité pour les sites correspondant à vos critères de recherche avec la fonction `regexp_like`, puis les classe par ordre décroissant.

```
WITH dataset AS (
  SELECT ARRAY[
    CAST(
      ROW('aws.amazon.com', ROW(ARRAY[
          MAP(ARRAY['term', 'count'], ARRAY['bigdata', '10']),
          MAP(ARRAY['term', 'count'], ARRAY['serverless', '50']),
          MAP(ARRAY['term', 'count'], ARRAY['analytics', '82']),
          MAP(ARRAY['term', 'count'], ARRAY['iot', '74'])
      ])
      ) AS ROW(hostname VARCHAR, flaggedActivity ROW(flags ARRAY(MAP(VARCHAR,
 VARCHAR)) ))
    ),
    CAST(
      ROW('news.cnn.com', ROW(ARRAY[
        MAP(ARRAY['term', 'count'], ARRAY['politics', '241']),
        MAP(ARRAY['term', 'count'], ARRAY['technology', '211']),
        MAP(ARRAY['term', 'count'], ARRAY['serverless', '25']),
        MAP(ARRAY['term', 'count'], ARRAY['iot', '170'])
      ])
      ) AS ROW(hostname VARCHAR, flaggedActivity ROW(flags ARRAY(MAP(VARCHAR,
 VARCHAR)) ))
    ),
    CAST(
      ROW('netflix.com', ROW(ARRAY[
        MAP(ARRAY['term', 'count'], ARRAY['cartoons', '1020']),
        MAP(ARRAY['term', 'count'], ARRAY['house of cards', '112042']),
        MAP(ARRAY['term', 'count'], ARRAY['orange is the new black', '342']),
        MAP(ARRAY['term', 'count'], ARRAY['iot', '4'])
      ])
      ) AS ROW(hostname VARCHAR, flaggedActivity ROW(flags ARRAY(MAP(VARCHAR,
 VARCHAR)) ))
    )
  ] AS items
),
sites AS (
  SELECT sites.hostname, sites.flaggedactivity
  FROM dataset, UNNEST(items) t(sites)
)
```

```
SELECT hostname, array_agg(flags['term']) AS terms, SUM(CAST(flags['count'] AS
 INTEGER)) AS total
FROM sites, UNNEST(sites.flaggedActivity.flags) t(flags)
WHERE regexp_like(flags['term'], 'politics|bigdata')
GROUP BY (hostname)
ORDER BY total DESC
```

Cette requête renvoie deux sites :

```
+-------------------------------------+
| hostname       | terms    | total  |
+----------------+------------------+
| news.cnn.com   | politics |  241   |
+----------------+------------------+
| aws.amazon.com | bigdata |  10    |
+----------------+------------------+
```

Interrogation de données géospatiales

Les données géospatiales contiennent des identifiants qui spécifient la position géographique d'un objet. Les rapports météo, les indications de direction, les tweets intégrant des lieux géographiques, les emplacements de magasins et les lignes aériennes sont des exemples de ce type de données. Les données géospatiales jouent un rôle important dans les analyses métier, la création de rapports et les prévisions.

Les identifiants géographiques, tels que la latitude et la longitude, vous permettent de convertir n'importe quelle adresse postale en un ensemble de coordonnées géographiques.

Rubriques

Qu'est-ce qu'une requête géospatiale ?

Les requêtes géospatiales sont des types de requête SQL spécialisés pris en charge dans Athena. Les aspects suivants les différencient des requêtes SQL non spatiales :

- Elles utilisent les types de données géométriques spécialisées suivantes : `point`, `line`, `multiline`, `polygon` et `multipolygon`.
- Elles expriment des relations entre les types de données géométriques, tels que `distance`, `equals`, `crosses`, `touches`, `overlaps`, `disjoint` etc.

À l'aide des requêtes géospatiales dans Athena, vous pouvez exécuter les opérations suivantes, ainsi que d'autres opérations similaires :

- Trouver la distance entre deux points.
- Vérifier si une zone (polygone) en contient une autre.
- Vérifier si une ligne croise ou touche une autre ligne ou un autre polygone.

Par exemple, pour obtenir un type de données de géométrie `point` à partir de valeurs de type `double` pour les coordonnées géographiques du Mont Rainier dans Athena, utilisez la fonction géospatiale `ST_Point (longitude, latitude)`, comme dans l'exemple suivant.

```
ST_Point(-121.7602, 46.8527)
```

Formats de données en entrée et types de données géométriques

Pour utiliser les fonctions géospatiales dans Athena, saisissez vos données au format WKT ou utilisez le JSON Hive. SerDe Vous pouvez également utiliser les types de données de géométrie pris en charge dans Athena.

Format de données en entrée

Pour traiter les requêtes géospatiales, Athena prend en charge les données d'entrée dans les formats suivants :

- WKT (Well-Known Text). Dans Athena, WKT est représenté en tant que type de données `varchar(x)` ou `string`.
- Données spatiales codées JSON. Pour analyser des fichiers JSON contenant des données géospatiales et créer des tables pour ceux-ci, Athena utilise le Hive JSON. SerDe Pour plus d'informations sur son utilisation SerDe dans Athena, consultez. SerDe bibliothèques JSON

Types de données géométriques

Pour traiter les requêtes géospatiales, Athena prend en charge les types de données de géométrie spécialisés suivants :

- `point`
- `line`
- `polygon`
- `multiline`
- `multipolygon`

Fonctions géospatiales prises en charge

Les fonctions géospatiales disponibles dans Athena dépendent de la version du moteur que vous utilisez.

- Pour plus d'informations sur les fonctions géospatiales de la version 3 du moteur Athena, veuillez consulter la rubrique Geospatial functions dans la documentation de Trino.
- Pour une liste des modifications de noms de fonctions et des nouvelles fonctions à partir de la version 2 du moteur Athena, veuillez consulter la rubrique Modifications des noms des fonctions géospatiales et nouvelles fonctions dans la version 2 du moteur Athena.

Pour plus d'informations sur la gestion des versions du moteur Athena, voir Gestion des versions du moteur Athena.

Rubriques

- Fonctions géospatiales de la version 3 du moteur Athena
- Fonctions géospatiales de la version 2 du moteur Athena

Fonctions géospatiales de la version 3 du moteur Athena

Pour plus d'informations sur les fonctions géospatiales de la version 3 du moteur Athena, veuillez consulter la rubrique Geospatial functions dans la documentation de Trino.

Fonctions géospatiales de la version 2 du moteur Athena

Cette rubrique répertorie les fonctions géospatiales ESRI qui sont prises en charge à partir de la version 2 du moteur Athena. Pour plus d'informations sur les versions du moteur Athena, voir Gestion des versions du moteur Athena.

Modifications apportées à la version 2 du moteur Athena

- Les types d'entrée et de sortie de certaines fonctions ont changé. En particulier, le type `VARBINARY` n'est plus directement pris en charge pour l'entrée. Pour plus d'informations, consultez Modifications apportées aux fonctions géospatiales.
- Les noms de certaines fonctions géospatiales ont changé. Pour plus d'informations, consultez Modifications des noms des fonctions géospatiales dans la version 2 du moteur Athena.
- De nouvelles fonctions ont été ajoutées. Pour plus d'informations, consultez Nouvelles fonctions géospatiales de la version 2 du moteur Athena.

Athena prend en charge les types de fonctions géospatiales suivants :

- Fonctions de constructeur
- fonctions de relations géospatiales
- fonctions d'opérations
- Fonctions d'accesseur
- Fonctions d'agrégation
- Fonctions Bing Tile

Fonctions de constructeur

Utilisez des fonctions de constructeur pour obtenir des représentations binaires des types de données géométriques `point`, `line` ou `polygon`. Vous pouvez également utiliser ces fonctions pour convertir des données binaires en texte et obtenir des valeurs binaires pour les données géométriques exprimées au format WKT (well-known text).

`ST_AsBinary(geometry)`

Renvoie un type de données varbinary qui contient la représentation WKB de la géométrie spécifiée. Exemple :

```
SELECT ST_AsBinary(ST_Point(-158.54, 61.56))
```

ST_AsText(geometry)

Convertit chacun des types de données géométriques spécifiés en texte. Renvoie une valeur dans un type de données varchar, qui est une représentation WKT du type de données de géométrie. Exemple :

```
SELECT ST_AsText(ST_Point(-158.54, 61.56))
```

ST_GeomAsLegacyBinary(geometry)

Renvoie un varbinary hérité à partir de la géométrie spécifiée. Exemple :

```
SELECT ST_GeomAsLegacyBinary(ST_Point(-158.54, 61.56)
```

ST_GeometryFromText(varchar)

Convertit un texte au format WKT en un type de données de géométrie. Renvoie une valeur dans un type de données de géométrie. Exemple :

```
SELECT ST_GeometryFromText(ST_AsText(ST_Point(1, 2)))
```

ST_GeomFromBinary(varbinary)

Renvoie un objet de type géométrique à partir d'une représentation WKB. Exemple :

```
SELECT ST_GeomFromBinary(ST_AsBinary(ST_Point(-158.54, 61.56)))
```

ST_GeomFromLegacyBinary(varbinary)

Renvoie un objet de type géométrique à partir d'un type varbinary hérité. Exemple :

```
SELECT ST_GeomFromLegacyBinary(ST_GeomAsLegacyBinary(ST_Point(-158.54, 61.56)))
```

ST_LineFromText(varchar)

Renvoie une valeur dans un type de données de géométrie `line`. Exemple :

```
SELECT ST_Line('linestring(1 1, 2 2, 3 3)')
```

ST_LineString(array(point))

Renvoie un type géométrique `LineString` formé à partir d'un tableau de types géométriques ponctuels. S'il y a moins de deux points non vides dans le tableau spécifié, une chaîne `LineString` vide est renvoyée. Renvoie une exception si un élément du tableau est nul, vide ou identique au précédent. Il est possible que la géométrie renvoyée ne soit pas simple. En fonction de l'entrée spécifiée, la géométrie renvoyée peut s'auto-intersecter ou contenir des sommets en double. Exemple :

```
SELECT ST_LineString(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_MultiPoint(array(point))

Renvoie un objet géométrique `MultiPoint` formé à partir des points spécifiés. Renvoie un caractère nul si le tableau spécifié est vide. Renvoie une exception si un élément du tableau est nul ou vide. Il est possible que la géométrie renvoyée ne soit pas simple et qu'elle contienne des points en double si le tableau spécifié en contient. Exemple :

```
SELECT ST_MultiPoint(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_Point(double, double)

Renvoie un objet `point` de type géométrique. Pour les valeurs de données d'entrée de cette fonction, utilisez des valeurs géométriques, telles que les valeurs du système de coordonnées cartésiennes en UTM (Universal Transverse Mercator) ou des unités de cartes géographiques (longitude et latitude) en degrés décimaux. Les valeurs de latitude et de longitude utilisent le système géodésique mondial, également appelé WGS 1984 ou EPSG:4326. WGS 1984 est le système de coordonnées utilisé par le GPS (Global Positioning System).

Par exemple, dans la notation suivante, les coordonnées cartographiques sont spécifiées dans la longitude et la latitude, et la valeur `.072284`, qui est la distance tampon, est spécifiée dans les unités angulaires en degrés décimaux :

```
SELECT ST_Buffer(ST_Point(-74.006801, 40.705220), .072284)
```

Syntaxe :

```
SELECT ST_Point(longitude, latitude) FROM earthquakes LIMIT 1
```

L'exemple ci-dessous utilise des coordonnées spécifiques de latitude et de longitude :

```
SELECT ST_Point(-158.54, 61.56)
FROM earthquakes
LIMIT 1
```

L'exemple suivant utilise des coordonnées spécifiques de latitude et de longitude :

```
SELECT ST_Point(-74.006801, 40.705220)
```

L'exemple suivant utilise la fonction `ST_AsText` pour obtenir la géométrie à partir de WKT :

```
SELECT ST_AsText(ST_Point(-74.006801, 40.705220)) AS WKT
```

ST_Polygon(varchar)

En utilisant la séquence des ordonnées fournies dans le sens horaire, de gauche à droite, on obtient un type de données de géométrie `polygon`. À partir de la version 2 du moteur Athena, seuls les polygones sont acceptés en entrée. Exemple :

```
SELECT ST_Polygon('polygon ((1 1, 1 4, 4 4, 4 1))')
```

to_geometry(sphericalGeography)

Renvoie un objet géométrique à partir de l'objet géographique sphérique spécifié. Exemple :

```
SELECT to_geometry(to_spherical_geography(ST_Point(-158.54, 61.56)))
```

to_spherical_geography(geometry)

Renvoie un objet géographique sphérique à partir de la géométrie spécifiée. Utilisez cette fonction pour convertir un objet géométrique en un objet géométrique sphérique sur la sphère du rayon de la Terre. La fonction ne peut être utilisée que sur les géométries `POINT`, `MULTIPOINT`, `LINESTRING`, `MULTILINESTRING`, `POLYGON` et `MULTIPOLYGON` définies dans un espace 2D ou une `GEOMETRYCOLLECTION` de ces géométries. Pour chaque point de la géométrie spécifiée, la fonction vérifie que `point.x` se trouve dans `[-180.0, 180.0]` et que `point.y` se trouve dans `[-90.0, 90.0]`. La fonction utilise ces points comme degrés de longitude et de latitude pour construire la forme du résultat `sphericalGeography`.

Exemple :

```
SELECT to_spherical_geography(ST_Point(-158.54, 61.56))
```

fonctions de relations géospatiales

Les fonctions suivantes expriment des relations entre deux géométries différentes que vous spécifiez en entrée et renvoient des résultats de type `boolean`. L'ordre dans lequel vous spécifiez la paire des géométries a de l'importance : la première valeur de géométrie est appelée géométrie de gauche et la deuxième, géométrie de droite.

Ces fonctions renvoient :

- `TRUE` si et seulement si la relation décrite par la fonction est remplie.
- `FALSE` si et seulement si la relation décrite par la fonction n'est pas remplie.

ST_Contains(geometry, geometry)

Renvoie `TRUE` si et seulement si la géométrie de gauche contient la géométrie de droite. Exemples :

```
SELECT ST_Contains('POLYGON((0 2,1 1,0 -1,0 2))', 'POLYGON((-1 3,2 1,0 -3,-1 3))')
```

```
SELECT ST_Contains('POLYGON((0 2,1 1,0 -1,0 2))', ST_Point(0, 0))
```

```
SELECT ST_Contains(ST_GeometryFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeometryFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'))
```

ST_Crosses(geometry, geometry)

Renvoie `TRUE` si et seulement si la géométrie de gauche croise la géométrie de droite. Exemple :

```
SELECT ST_Crosses(ST_Line('linestring(1 1, 2 2 )'), ST_Line('linestring(0 1, 2 2)'))
```

ST_Disjoint(geometry, geometry)

Renvoie `TRUE` si et seulement si l'intersection des géométries de gauche et de droite est vide. Exemple :

```
SELECT ST_Disjoint(ST_Line('linestring(0 0, 0 1)'), ST_Line('linestring(1 1, 1 0)'))
```

ST_Equals(geometry, geometry)

Renvoie TRUE si et seulement si la géométrie de gauche est égale à la géométrie de droite. Exemple :

```
SELECT ST_Equals(ST_Line('linestring( 0 0, 1 1)'), ST_Line('linestring(1 3, 2 2)'))
```

ST_Intersects(geometry, geometry)

Renvoie TRUE si et seulement si la géométrie de gauche coupe la géométrie de droite. Exemple :

```
SELECT ST_Intersects(ST_Line('linestring(8 7, 7 8)'), ST_Polygon('polygon((1 1, 4 1, 4
 4, 1 4))'))
```

ST_Overlaps(geometry, geometry)

Renvoie TRUE si et seulement si la géométrie de gauche chevauche la géométrie de droite. Exemple :

```
SELECT ST_Overlaps(ST_Polygon('polygon((2 0, 2 1, 3 1))'), ST_Polygon('polygon((1 1, 1
 4, 4 4, 4 1))'))
```

ST_Relate(geometry, geometry, varchar)

Renvoie TRUE si et seulement si la géométrie de gauche a la relation DE-9IM (Dimensionally Extended nine-Intersection Model) spécifiée avec la géométrie de droite. La troisième entrée (`varchar`) prend la relation. Exemple :

```
SELECT ST_Relate(ST_Line('linestring(0 0, 3 3)'), ST_Line('linestring(1 1, 4 4)'),
 'T********')
```

ST_Touches(geometry, geometry)

Renvoie TRUE si et seulement si la géométrie de gauche touche la géométrie de droite.

Exemple :

```
SELECT ST_Touches(ST_Point(8, 8), ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

ST_Within(geometry, geometry)

Renvoie TRUE si et seulement si la géométrie de gauche est à l'intérieur de la géométrie de droite.

Exemple :

```
SELECT ST_Within(ST_Point(8, 8), ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

fonctions d'opérations

Utilisez des fonctions d'opérations pour exécuter des opérations sur des valeurs de type de données géométrique. Par exemple, vous pouvez obtenir les limites d'un type de données géométrique unique : intersections entre deux types de données géométriques ou différence entre les géométries de gauche et de droite, où chacune des géométries est du même type de données géométrique, ou un tampon ou anneau extérieur autour d'un type de données géométrique particulier.

geometry_union(array(geometry))

Renvoie une géométrie qui représente l'union de l'ensemble des points des géométries spécifiées. Exemple :

```
SELECT geometry_union(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_Boundary(geometry)

Prend en entrée l'un des types de données géométriques et renvoie le type de données de géométrie `boundary`.

Exemples :

```
SELECT ST_Boundary(ST_Line('linestring(0 1, 1 0)')))
```

```
SELECT ST_Boundary(ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

ST_Buffer(geometry, double)

Prend en entrée l'un des types de données géométriques, comme un point, une ligne, un polygone, une multiligne ou multipolygone, et une distance comme type `double`). Renvoie le type de données de géométrie mis en mémoire tampon par la distance (ou le rayon) spécifiée. Exemple :

```
SELECT ST_Buffer(ST_Point(1, 2), 2.0)
```

Dans l'exemple suivant, les coordonnées cartographiques sont spécifiées dans la longitude et la latitude, et la valeur `.072284`, qui est la distance tampon, est spécifiée dans les unités angulaires en degrés décimaux :

```
SELECT ST_Buffer(ST_Point(-74.006801, 40.705220), .072284)
```

ST_Difference(geometry, geometry)

Renvoie une géométrie de la différence entre la géométrie de gauche et la géométrie de droite. Exemple :

```
SELECT ST_AsText(ST_Difference(ST_Polygon('polygon((0 0, 0 10, 10 10, 10 0))'),
 ST_Polygon('polygon((0 0, 0 5, 5 5, 5 0))')))
```

ST_Envelope(geometry)

Accepte en tant qu'entrée les types de données géométriques `line`, `polygon`, `multiline` et `multipolygon`. Ne prend pas en charge le type de données géométrique `point`. Renvoie l'enveloppe sous forme de géométrie, l'enveloppe étant un rectangle autour du type spécifié de données de géométrie. Exemples :

```
SELECT ST_Envelope(ST_Line('linestring(0 1, 1 0)'))
```

```
SELECT ST_Envelope(ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

ST_EnvelopeAsPts(geometry)

Renvoie un tableau de deux points qui représentent les coins inférieurs gauche et supérieur droit du polygone rectangulaire d'une géométrie. Renvoie un caractère nul si la géométrie spécifiée est vide. Exemple :

```
SELECT ST_EnvelopeAsPts(ST_Point(-158.54, 61.56))
```

ST_ExteriorRing(geometry)

Renvoie la géométrie de l'anneau extérieur du type d'entrée `polygon`. À partir de la version 2 du moteur Athena, les polygones sont les seules géométries acceptées en entrée. Exemples :

```
SELECT ST_ExteriorRing(ST_Polygon(1,1, 1,4, 4,1))
```

```
SELECT ST_ExteriorRing(ST_Polygon('polygon ((0 0, 8 0, 0 8, 0 0), (1 1, 1 5, 5 1, 1
 1))'))
```

ST_Intersection(geometry, geometry)

Renvoie la géométrie de l'intersection de la géométrie gauche et de la géométrie droite. Exemples :

```
SELECT ST_Intersection(ST_Point(1,1), ST_Point(1,1))
```

```
SELECT ST_Intersection(ST_Line('linestring(0 1, 1 0)'), ST_Polygon('polygon((1  1, 1
 4, 4  4, 4 1))'))
```

```
SELECT ST_AsText(ST_Intersection(ST_Polygon('polygon((2 0, 2 3, 3 0))'),
 ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))')))
```

ST_SymDifference(geometry, geometry)

Renvoie la géométrie de la différence géométriquement symétrique entre la géométrie de gauche et la géométrie de droite. Exemple :

```
SELECT ST_AsText(ST_SymDifference(ST_Line('linestring(0 2, 2 2)'),
 ST_Line('linestring(1 2, 3 2)')))
```

ST_Union(geometry, geometry)

Renvoie un type de données de géométrie qui représente l'union des ensembles de points des géométries spécifiées. Exemple :

```
SELECT ST_Union(ST_Point(-158.54, 61.56),ST_LineString(array[ST_Point(1,2),
 ST_Point(3,4)]))
```

Fonctions d'accesseur

Les fonctions d'accesseur s'avèrent utiles pour obtenir des valeurs dans des types `varchar`, `bigint` ou `double` de différents types de données `geometry`, où `geometry` est l'un des types de données de géométrie pris en charge dans Athena : `point`, `line`, `polygon`, `multiline` et

`multipolygon`. Par exemple, vous pouvez obtenir une surface d'un type de données géométrique `polygon`, les valeurs X et Y maximum et minimum pour un type de données géométrique spécifié, la longueur d'un type de données `line` ou le nombre de points dans un type de données géométrique indiqué.

geometry_invalid_reason(geometry)

Renvoie, dans un type de données varchar, la raison pour laquelle la géométrie spécifiée n'est pas valide ou pas simple. Si la géométrie spécifiée n'est ni valide ni simple, renvoie la raison pour laquelle elle n'est pas valide. Si la géométrie spécifiée est valide et simple, renvoie un caractère nul. Exemple :

```
SELECT geometry_invalid_reason(ST_Point(-158.54, 61.56))
```

great_circle_distance(latitude1, longitude1, latitude2, longitude2)

Renvoie, en double, la distance orthodromique entre deux points de la surface de la Terre en kilomètres. Exemple :

```
SELECT great_circle_distance(36.12, -86.67, 33.94, -118.40)
```

line_locate_point(lineString, point)

Renvoie un double entre 0 et 1 qui représente l'emplacement du point le plus proche du point spécifié sur la chaîne de lignes spécifiée, en tant que fraction de la longueur totale de la ligne 2d.

Renvoie un caractère nul si la chaîne de lignes ou le point spécifié est vide ou nul. Exemple :

```
SELECT line_locate_point(ST_GeometryFromText('LINESTRING (0 0, 0 1)'), ST_Point(0,
 0.2))
```

simplify_geometry(geometry, double)

Utilise l'amer-douglas-peucker algorithme R pour renvoyer un type de données de géométrie qui est une version simplifiée de la géométrie spécifiée. Évite de créer des géométries dérivées (en particulier des polygones) non valides. Exemple :

```
SELECT simplify_geometry(ST_GeometryFromText('POLYGON ((1 0, 2 1, 3 1, 3 1, 4 1, 1
 0))'), 1.5)
```

ST_Area(geometry)

Accepte en tant qu'entrée un type de données géométrique et renvoie une surface dans un type `double`. Exemple :

```
SELECT ST_Area(ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))'))
```

ST_Centroid(geometry)

Accepte en tant qu'entrée un type géométrique de données `polygon` et renvoie un type de données de géométrie `point` qui est le centre de l'enveloppe du polygone. Exemples :

```
SELECT ST_Centroid(ST_GeometryFromText('polygon ((0 0, 3 6, 6 0, 0 0))'))
```

```
SELECT ST_AsText(ST_Centroid(ST_Envelope(ST_GeometryFromText('POINT (53 27)'))))
```

ST_ConvexHull(geometry)

Renvoie un type de données de géométrie qui est la plus petite géométrie convexe qui englobe toutes les géométries dans l'entrée spécifiée. Exemple :

```
SELECT ST_ConvexHull(ST_Point(-158.54, 61.56))
```

ST_CoordDim(geometry)

Prend en entrée l'un des type de données de géométrie pris en charge et renvoie le nombre de composants de coordonnées dans le type `tinyint`. Exemple :

```
SELECT ST_CoordDim(ST_Point(1.5,2.5))
```

ST_Dimension(geometry)

Accepte en tant qu'entrée l'un des types de données géométriques pris en charge et renvoie la dimension spatiale d'une géométrie en type `tinyint`. Exemple :

```
SELECT ST_Dimension(ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))'))
```

ST_Distance(geometry, geometry)

Renvoie, en fonction de la référence spatiale, un double contenant la distance cartésienne minimale bidimensionnelle entre deux géométries en unités projetées. À partir de la version 2 du moteur Athena, renvoie un caractère nul si l'une des entrées est une géométrie vide. Exemple :

```
SELECT ST_Distance(ST_Point(0.0,0.0), ST_Point(3.0,4.0))
```

ST_Distance(sphericalGeography, sphericalGeography)

Renvoie, sous forme de double, la distance orthodromique entre deux points géographiques sphériques en mètres. Exemple :

```
SELECT ST_Distance(to_spherical_geography(ST_Point(61.56,
 -86.67)),to_spherical_geography(ST_Point(61.56, -86.68)))
```

ST_EndPoint(geometry)

Renvoie le dernier point d'un type de données de géométrie `line` dans un type de données de géométrie `point`. Exemple :

```
SELECT ST_EndPoint(ST_Line('linestring(0 2, 2 2)'))
```

ST_Geometries(geometry)

Renvoie un tableau des géométries dans la collection spécifiée. Si la géométrie spécifiée n'est pas une géométrie multiple, elle renvoie un tableau à un élément. Si la géométrie spécifiée est vide, un caractère nul est renvoyé.

Par exemple, avec un objet `MultiLineString`, `ST_Geometries` crée un tableau d'objets `LineString`. Avec un objet `GeometryCollection`, `ST_Geometries` renvoie un tableau non aplati de ses constituants. Exemple :

```
SELECT ST_Geometries(GEOMETRYCOLLECTION(MULTIPOINT(0 0, 1 1),
 GEOMETRYCOLLECTION(MULTILINESTRING((2 2, 3 3)))))
```

Résultat:

```
array[MULTIPOINT(0 0, 1 1),GEOMETRYCOLLECTION(MULTILINESTRING((2 2, 3 3)))]
```

ST_GeometryN(geometry, index)

Renvoie, en tant que type de données de géométrie, l'élément géométrique à un index entier spécifié. Les indices commencent à 1. Si la géométrie spécifiée est une collection de géométries (par exemple, un objet `GEOMETRYCOLLECTION` ou `MULTI*`), renvoie la géométrie à l'index spécifié. Si l'index spécifié est inférieur à 1 ou supérieur au nombre total d'éléments de la collection, renvoie un caractère nul. Pour obtenir le nombre total d'éléments, utilisez `ST_NumGeometries`. Les géométries singulières (par exemple, `POINT`, `LINESTRING`, ou `POLYGON`), sont traitées comme des collections d'un élément. Les géométries vides sont traitées comme des collections vides. Exemple :

```
SELECT ST_GeometryN(ST_Point(-158.54, 61.56),1)
```

ST_GeometryType(geometry)

Renvoie, sous la forme d'un varchar, le type de la géométrie. Exemple :

```
SELECT ST_GeometryType(ST_Point(-158.54, 61.56))
```

ST_InteriorRingN(geometry, index)

Renvoie l'élément anneau intérieur à l'index spécifié (les indices commencent à 1). Si l'indice donné est inférieur à 1 ou supérieur au nombre total d'anneaux intérieurs dans la géométrie spécifiée, renvoie un caractère nul. Renvoie une erreur si la géométrie spécifiée n'est pas un polygone. Pour obtenir le nombre total d'éléments, utilisez `ST_NumInteriorRing`. Exemple :

```
SELECT ST_InteriorRingN(st_polygon('polygon ((0 0, 1 0, 1 1, 0 1, 0 0))'),1)
```

ST_InteriorRings(geometry)

Renvoie un tableau de géométrie de tous les anneaux intérieurs trouvés dans la géométrie spécifiée, ou un tableau vide si le polygone n'a pas d'anneaux intérieurs. Si la géométrie spécifiée est vide, un caractère nul est renvoyé. Si la géométrie spécifiée n'est pas un polygone, renvoie une erreur. Exemple :

```
SELECT ST_InteriorRings(st_polygon('polygon ((0 0, 1 0, 1 1, 0 1, 0 0))'))
```

ST_IsClosed(geometry)

Prend comme entrée uniquement et les types de données de géométrie `line` et `multiline`. Renvoie `TRUE` (type `boolean`) si et seulement si la ligne est fermée. Exemple :

```
SELECT ST_IsClosed(ST_Line('linestring(0 2, 2 2)'))
```

ST_IsEmpty(geometry)

Prend comme entrée uniquement et les types de données de géométrie `line` et `multiline`. Renvoie TRUE (type `boolean`) si et seulement si la géométrie spécifiée est vide, autrement dit, si les valeurs de début et de fin de `line` coïncident. Exemple :

```
SELECT ST_IsEmpty(ST_Point(1.5, 2.5))
```

ST_IsRing(geometry)

Renvoie TRUE (type `boolean`) si et seulement si le type `line` est fermé et simple. Exemple :

```
SELECT ST_IsRing(ST_Line('linestring(0 2, 2 2)'))
```

ST_IsSimple(geometry)

Renvoie VRAI si la géométrie spécifiée ne comporte pas de points géométriques anormaux (par exemple, une auto-intersection ou une auto-tangence). Pour déterminer pourquoi la géométrie n'est pas simple, utilisez `geometry_invalid_reason()`. Exemple :

```
SELECT ST_IsSimple(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]))
```

ST_IsValid(geometry)

Renvoie VRAI si et seulement si la géométrie spécifiée est bien formée. Pour déterminer pourquoi la géométrie n'est pas bien formée, utilisez `geometry_invalid_reason()`. Exemple :

```
SELECT ST_IsValid(ST_Point(61.56, -86.68))
```

ST_Length(geometry)

Renvoie la longueur de `line` en type `double`. Exemple :

```
SELECT ST_Length(ST_Line('linestring(0 2, 2 2)'))
```

ST_NumGeometries(geometry)

Renvoie, sous la forme d'un entier, le nombre de géométries dans la collection. Si la géométrie est une collection de géométries (par exemple, un objet `GEOMETRYCOLLECTION` ou `MULTI*`), renvoie le nombre de géométries. Les géométries simples renvoient 1 ; les géométries vides renvoient 0. Une géométrie vide dans un objet `GEOMETRYCOLLECTION` compte comme une géométrie. Par exemple, l'exemple suivant donne la valeur 1 :

```
ST_NumGeometries(ST_GeometryFromText('GEOMETRYCOLLECTION(MULTIPOINT EMPTY)'))
```

ST_NumInteriorRing(geometry)

Renvoie le nombre d'anneaux intérieurs dans la géométrie `polygon` en type `bigint`. Exemple :

```
SELECT ST_NumInteriorRing(ST_Polygon('polygon ((0 0, 8 0, 0 8, 0 0), (1 1, 1 5, 5 1, 1
 1))'))
```

ST_NumPoints(geometry)

Renvoie le nombre de points de la géométrie en type `bigint`. Exemple :

```
SELECT ST_NumPoints(ST_Point(1.5, 2.5))
```

ST_PointN(lineString, index)

Renvoie, en tant que données de géométrie de type point, le sommet de la chaîne de lignes spécifiée à l'index entier spécifié. Les indices commencent à 1. Si l'index donné est inférieur à 1 ou supérieur au nombre total d'éléments de la collection, renvoie un caractère nul. Pour obtenir le nombre total d'éléments, utilisez `ST_NumPoints`. Exemple :

```
SELECT ST_PointN(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]),1)
```

ST_Points(geometry)

Renvoie un tableau de points de l'objet géométrique de la chaîne de lignes spécifiée. Exemple :

```
SELECT ST_Points(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]))
```

ST_StartPoint(geometry)

Renvoie le premier point d'un type de données de géométrie `line` dans un type de données de géométrie `point`. Exemple :

```
SELECT ST_StartPoint(ST_Line('linestring(0 2, 2 2)'))
```

ST_X(point)

Renvoie la coordonnée X d'un point en type `double`. Exemple :

```
SELECT ST_X(ST_Point(1.5, 2.5))
```

ST_XMax(geometry)

Renvoie la coordonnée X maximale d'une géométrie en type `double`. Exemple :

```
SELECT ST_XMax(ST_Line('linestring(0 2, 2 2)'))
```

ST_XMin(geometry)

Renvoie la coordonnée X minimale d'une géométrie en type `double`. Exemple :

```
SELECT ST_XMin(ST_Line('linestring(0 2, 2 2)'))
```

ST_Y(point)

Renvoie la coordonnée Y d'un point en type `double`. Exemple :

```
SELECT ST_Y(ST_Point(1.5, 2.5))
```

ST_YMax(geometry)

Renvoie la coordonnée Y maximale d'une géométrie en type `double`. Exemple :

```
SELECT ST_YMax(ST_Line('linestring(0 2, 2 2)'))
```

ST_YMin(geometry)

Renvoie la coordonnée Y minimale d'une géométrie en type `double`. Exemple :

```
SELECT ST_YMin(ST_Line('linestring(0 2, 2 2)'))
```

Fonctions d'agrégation

convex_hull_agg(geometry)

Renvoie la géométrie convexe minimale qui englobe toutes les géométries transmises en entrée.

geometry_union_agg(geometry)

Renvoie une géométrie qui représente l'union des ensembles de points de toutes les géométries transmises en entrée.

Fonctions Bing Tile

Les fonctions suivantes permettent de convertir entre les géométries et les tuiles dans le Système de tuiles Bing de Microsoft.

bing_tile(x, y, zoom_level)

Renvoie un objet tuile Bing à partir de coordonnées entières `x` et `y` et du niveau de zoom spécifié. Le niveau de zoom doit être un entier compris entre 1 et 23. Exemple :

```
SELECT bing_tile(10, 20, 12)
```

bing_tile(quadKey)

Renvoie un objet tuile Bing à partir d'une quadkey. Exemple :

```
SELECT bing_tile(bing_tile_quadkey(bing_tile(10, 20, 12)))
```

bing_tile_at(latitude, longitude, zoom_level)

Renvoie un objet tuile Bing à la latitude, la longitude et au niveau de zoom spécifiés. La latitude doit être comprise entre -85,05112878 et 85,05112878. La longitude doit être comprise entre -180 et 180. Les valeurs `latitude` et `longitude` doivent être un entier `double` et `zoom_level`. Exemple :

```
SELECT bing_tile_at(37.431944, -122.166111, 12)
```

bing_tiles_around(latitude, longitude, zoom_level)

Renvoie un tableau des tuiles Bing qui entourent le point de latitude et de longitude spécifié au niveau de zoom spécifié. Exemple :

```
SELECT bing_tiles_around(47.265511, -122.465691, 14)
```

bing_tiles_around(latitude, longitude, zoom_level, radius_in_km)

Renvoie, au niveau de zoom spécifié, un tableau de tuiles Bing. Le tableau contient l'ensemble minimal de tuiles Bing qui couvre un cercle du rayon spécifié en kilomètres autour de la latitude et de la longitude spécifiées. Les valeurs `latitude`, `longitude` et `radius_in_km` sont `double` ; le niveau de zoom est un `integer`. Exemple :

```
SELECT bing_tiles_around(37.8475, 112.596667, 10, .5)
```

bing_tile_coordinates(tile)

Renvoie les coordonnées `x` et `y` de la tuile Bing spécifiée. Exemple :

```
SELECT bing_tile_coordinates(bing_tile_at(37.431944, -122.166111, 12))
```

bing_tile_polygon(tile)

Renvoie la représentation polygonale de la tuile Bing spécifiée. Exemple :

```
SELECT bing_tile_polygon(bing_tile_at(47.265511, -122.465691, 4))
```

bing_tile_quadkey(tile)

Renvoie la quadkey de la tuile Bing spécifiée. Exemple :

```
SELECT bing_tile_quadkey(bing_tile(52, 143, 10))
```

bing_tile_zoom_level(tile)

Renvoie le niveau de zoom de la tuile Bing spécifiée sous la forme d'un entier. Exemple :

```
SELECT bing_tile_zoom_level(bing_tile(52, 143, 10))
```

`geometry_to_bing_tiles(geometry, zoom_level)`

Renvoie l'ensemble minimal de tuiles Bing qui couvre entièrement la géométrie spécifiée au niveau de zoom spécifié. Les niveaux de zoom de 1 à 23 sont pris en charge. Exemple :

```
SELECT geometry_to_bing_tiles(ST_Point(61.56, 58.54), 10)
```

Modifications des noms des fonctions géospatiales et nouvelles fonctions dans la version 2 du moteur Athena

Cette section répertorie les modifications apportées aux noms des fonctions géospatiales et les fonctions géospatiales qui sont nouvelles dans la version 2 du moteur Athena. Pour plus d'informations sur les autres modifications à partir de la version 2 du moteur Athena, veuillez consulter la rubrique Version 2 du moteur Athena.

Pour plus d'informations sur la gestion des versions du moteur Athena, voir Gestion des versions du moteur Athena.

Modifications des noms des fonctions géospatiales dans la version 2 du moteur Athena

Les noms des fonctions suivantes ont changé. Dans certains cas, les types d'entrée et de sortie ont également changé. Pour plus d'informations, consultez les liens correspondants.

Ancien nom de la fonction	Nom de fonction à partir de la version 2 du moteur Athena
st_coordinate_dimension	ST_ CoordDim
st_end_point	ST_ EndPoint
st_exterior_ring	ST_ ExteriorRing
st_interior_ring_number	ST_ NumInteriorRing
st_geometry_from_text	ST_ GeometryFromText
st_is_closed	ST_ IsClosed
st_is_empty	ST_ IsEmpty
st_is_ring	ST_ IsRing

Ancien nom de la fonction	Nom de fonction à partir de la version 2 du moteur Athena
st_max_x	ST_XMax
st_max_y	ST_YMax
st_min_x	ST_XMin
st_min_y	ST_YMin
st_point_number	ST_ NumPoints
st_start_point	ST_ StartPoint
st_symmetric_difference	ST_ SymDifference

Nouvelles fonctions géospatiales de la version 2 du moteur Athena

Les fonctions géospatiales suivantes sont nouvelles à partir de la version 2 du moteur Athena. Pour plus d'informations, consultez les liens correspondants.

Fonctions de constructeur

- ST_ AsBinary
- ST_ GeomAsLegacyBinary
- ST_ GeomFromBinary
- ST_ GeomFromLegacyBinary
- ST_ LineString
- ST_ MultiPoint
- to_geometry
- to_spherical_geography

Fonctions d'opérations

- geometry_union
- ST_ EnvelopeAsPts
- ST_Union

Fonctions d'accesseur

- geometry_invalid_reason
- great_circle_distance
- line_locate_point
- simplify_geometry
- ST_ ConvexHull
- ST_Distance (géographie sphérique)
- ST_Geometries
- ST_GeometryN
- ST_ GeometryType
- ST_N InteriorRing
- ST_ InteriorRings
- ST_ IsSimple
- ST_ IsValid
- ST_ NumGeometries
- ST_PointN
- ST_Points

Fonctions d'agrégation

- convex_hull_agg
- geometry_union_agg

Fonctions Bing tile

- bing_tile
- bing_tile (quadkey)
- bing_tile_at
- bing_tiles_around
- bing_tiles_around (rayon)
- bing_tile_coordinates

- bing_tile_polygon
- bing_tile_quadkey
- bing_tile_zoom_level
- geometry_to_bing_tiles

Exemples : requêtes géospatiales

Les exemples présentés dans cette rubrique créent deux tables à partir d'échantillons de données disponibles sur GitHub et interrogent les tables en fonction de ces données. Les exemples de données, qui sont à titre d'illustration seulement et dont l'exactitude n'est pas garantie, se trouvent dans les fichiers suivants :

- **`earthquakes.csv`** – Répertorie les tremblements de terre qui se sont produits en Californie. La table `earthquakes` d'exemple utilise des champs provenant de ces données.
- **`california-counties.json`** – Répertorie les données de comté de l'État de Californie au format GeoJSON conforme ESRI. Les données comprennent de nombreux champs tels que `AREA`, `PERIMETER`, `STATE`, `COUNTY` et `NAME`, mais la `counties` table d'exemple en utilise seulement deux : `Name` (chaîne) et `BoundaryShape` (binaire).

> **Note**
>
> Athena utilise le format `com.esri.json.hadoop.EnclosedEsriJsonInputFormat` pour convertir les données JSON au format binaire géospatial.

L'exemple de code suivant crée une table appelée `earthquakes` :

```
CREATE external TABLE earthquakes
(
 earthquake_date string,
 latitude double,
 longitude double,
 depth double,
 magnitude double,
 magtype string,
 mbstations string,
 gap string,
 distance string,
```

```
 rms string,
 source string,
 eventid string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION 's3://DOC-EXAMPLE-BUCKET/my-query-log/csv/';
```

L'exemple de code suivant crée une table appelée `counties` :

```
CREATE external TABLE IF NOT EXISTS counties
 (
 Name string,
 BoundaryShape binary
 )
ROW FORMAT SERDE 'com.esri.hadoop.hive.serde.EsriJsonSerDe'
STORED AS INPUTFORMAT 'com.esri.json.hadoop.EnclosedEsriJsonInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/my-query-log/json/';
```

L'exemple de requête suivant utilise la fonction `CROSS JOIN` sur les tables `counties` et `earthquake`. L'exemple utilise `ST_CONTAINS` pour interroger les comtés dont les limites incluent des zones sismiques, qui sont spécifiées avec `ST_POINT`. La requête regroupe ces comtés par nom, les ordonne par nombre et les renvoie en ordre décroissant.

> **Note**
>
> À partir de la version 2 du moteur Athena, les fonctions comme `ST_CONTAINS` ne prennent plus en charge le type `VARBINARY` comme entrée. Pour cette raison, l'exemple utilise la fonction `ST_GeomFromLegacyBinary(varbinary)` pour convertir la valeur binaire `boundaryshape` en une géométrie. Pour plus d'informations, voir Modifications apportées aux fonctions géospatiales dans la référence Version 2 du moteur Athena.

```
SELECT counties.name,
        COUNT(*) cnt
FROM counties
CROSS JOIN earthquakes
WHERE ST_CONTAINS (ST_GeomFromLegacyBinary(counties.boundaryshape),
 ST_POINT(earthquakes.longitude, earthquakes.latitude))
GROUP BY  counties.name
```

```
ORDER BY  cnt DESC
```

Cette requête renvoie :

```
+------------------------+
| name             | cnt |
+------------------------+
| Kern             | 36  |
+------------------------+
| San Bernardino   | 35  |
+------------------------+
| Imperial         | 28  |
+------------------------+
| Inyo             | 20  |
+------------------------+
| Los Angeles      | 18  |
+------------------------+
| Riverside        | 14  |
+------------------------+
| Monterey         | 14  |
+------------------------+
| Santa Clara      | 12  |
+------------------------+
| San Benito       | 11  |
+------------------------+
| Fresno           | 11  |
+------------------------+
| San Diego        | 7   |
+------------------------+
| Santa Cruz       | 5   |
+------------------------+
| Ventura          | 3   |
+------------------------+
| San Luis Obispo  | 3   |
+------------------------+
| Orange           | 2   |
+------------------------+
| San Mateo        | 1   |
+------------------------+
```

Ressources supplémentaires

Pour obtenir des exemples supplémentaires de requêtes géospatiales, veuillez consulter les billets de blog suivants :

- Étendez les requêtes géospatiales dans Amazon Athena avec des UDF et AWS Lambda
- Visualisez plus de 200 ans de données climatiques mondiales à l'aide d'Amazon Athena et Amazon. QuickSight
- Effectuer des requêtes OpenStreetMap avec Amazon Athena

Interrogation de JSON

Amazon Athena vous permet d'interroger des données codées en JSON, d'extraire des données du JSON imbriqué, de rechercher des valeurs et de déterminer la longueur et la taille des tableaux JSON. Pour apprendre les bases de l'interrogation de données JSON dans Athena, considérez les exemples de données planétaires suivants :

```
{name:"Mercury",distanceFromSun:0.39,orbitalPeriod:0.24,dayLength:58.65}
{name:"Venus",distanceFromSun:0.72,orbitalPeriod:0.62,dayLength:243.02}
{name:"Earth",distanceFromSun:1.00,orbitalPeriod:1.00,dayLength:1.00}
{name:"Mars",distanceFromSun:1.52,orbitalPeriod:1.88,dayLength:1.03}
```

Notez que chaque enregistrement (essentiellement chaque ligne du tableau) se trouve sur une ligne distincte. Pour interroger ces données JSON, vous pouvez utiliser une `CREATE TABLE` instruction comme celle-ci :

```
CREATE EXTERNAL TABLE `planets_json`(
  `name` string,
  `distancefromsun` double,
  `orbitalperiod` double,
  `daylength` double)
ROW FORMAT SERDE
  'org.openx.data.jsonserde.JsonSerDe'
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/json/'
```

Pour interroger les données, utilisez une SELECT instruction simple comme dans l'exemple suivant.

```
SELECT * FROM planets_json
```

Les résultats de la requête se présentent comme suit.

#	name	distance du soleil	période orbitale	durée du jour
1	Mercure	0,39	0,24	58,65
2	Vénus	0,72	0,62	243,02
3	Terre	1.0	1.0	1.0
4	Mars	1,52	1,88	1,03

Remarquez comment l'CREATE TABLEinstruction utilise leOpenX JSON SerDe, ce qui nécessite que chaque enregistrement JSON figure sur une ligne séparée. Si le JSON est dans un joli format d'impression, ou si tous les enregistrements se trouvent sur une seule ligne, les données ne seront pas lues correctement.

Pour interroger des données JSON dans un joli format d'impression, vous pouvez utiliser le JSON SerDe Amazon Ion Hive au lieu d'OpenX. SerDe Considérez les données précédentes stockées dans un joli format d'impression :

```
{
  name:"Mercury",
  distanceFromSun:0.39,
  orbitalPeriod:0.24,
  dayLength:58.65
}
{
  name:"Venus",
  distanceFromSun:0.72,
  orbitalPeriod:0.62,
  dayLength:243.02
}
{
  name:"Earth",
  distanceFromSun:1.00,
```

```
  orbitalPeriod:1.00,
  dayLength:1.00
}
{
  name:"Mars",
  distanceFromSun:1.52,
  orbitalPeriod:1.88,
  dayLength:1.03
}
```

Pour interroger ces données sans les reformater, vous pouvez utiliser une `CREATE TABLE` instruction comme celle-ci. Notez qu'au lieu de spécifier le code JSON OpenX SerDe, l'instruction spécifie. `STORED AS ION`

```
CREATE EXTERNAL TABLE `planets_ion`(
  `name` string,
  `distancefromsun` DECIMAL(10, 2),
  `orbitalperiod` DECIMAL(10, 2),
  `daylength` DECIMAL(10, 2))
STORED AS ION
LOCATION
  's3://DOC-EXAMPLE-BUCKET/json-ion/'
```

La requête `SELECT * FROM planets_ion` produit les mêmes résultats que précédemment. Pour plus d'informations sur la création de tables de cette manière à l'aide d'Amazon Ion Hive SerDe, consultezUtilisation de CREATE TABLE pour créer des tables Amazon Ion.

L'exemple de données JSON précédent ne contient pas de types de données complexes tels que des tableaux imbriqués ou des structures. Pour plus d'informations sur l'interrogation de données JSON imbriquées, consultez. Exemple : désérialisation des données JSON imbriquées

Rubriques

- Bonnes pratiques pour la lecture des données JSON
- Extraction de données JSON à partir de chaînes
- Recherche de valeurs dans les tableaux JSON
- Obtention de la longueur et de la taille de tableaux JSON
- Résolution des problèmes de requêtes JSON

Bonnes pratiques pour la lecture des données JSON

JavaScript La notation d'objet (JSON) est une méthode courante pour coder des structures de données sous forme de texte. De nombreux outils et applications produisent des données codées en JSON.

Dans Amazon Athena, vous pouvez créer des tables à partir de données externes et y inclure des données codées en JSON. Pour ce type de données source, utilisez Athena avec SerDe bibliothèques JSON.

Utilisez les conseils suivants pour lire les données encodées JSON :

- Choisissez le bon SerDe, un JSON SerDe natif ou un OpenX SerDe. `org.apache.hive.hcatalog.data.JsonSerDe` `org.openx.data.jsonserde.JsonSerDe` Pour plus d'informations, consultez SerDe bibliothèques JSON.
- Assurez-vous que chaque enregistrement encodé JSON est représenté sur une ligne distincte, et au non format d'impression (pretty-printed).

> **Note**
>
> Il SerDe s'attend à ce que chaque document JSON se trouve sur une seule ligne de texte sans aucun caractère de fin de ligne séparant les champs de l'enregistrement. Si le texte JSON est dans un joli format d'impression, vous pouvez recevoir un message d'erreur tel que HIVE_CURSOR_ERROR : Row is not a valid JSON Object ou HIVE_CURSOR_ERROR : : Unexpected JsonParseException end-of-input : expected close marker for OBJECT lorsque vous essayez d'interroger la table après l'avoir créée. Pour plus d'informations, consultez la section Fichiers de données JSON dans la SerDe documentation OpenX sur. GitHub

- Générez vos données encodées JSON dans des colonnes non sensibles à la casse.
- Fournissez une option pour ignorer les enregistrements incorrects, comme dans cet exemple.

```
CREATE EXTERNAL TABLE json_table (
  column_a string,
  column_b int
 )
 ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
 WITH SERDEPROPERTIES ('ignore.malformed.json' = 'true')
```

```
  LOCATION 's3://DOC-EXAMPLE-BUCKET/path/';
```

- Convertissez les champs en données sources ayant un schéma indéterminé en chaînes encodées JSON dans Athena.

Quand Athena crée des tables basées sur des données JSON, le service analyse les données en fonction du schéma existant et prédéfini. Cependant, toutes vos données peuvent ne pas avoir un schéma prédéfini. Pour simplifier la gestion des schémas dans de tels cas, il s'avère souvent utile de convertir les champs des données source ayant un schéma non déterminé en chaînes JSON dans Athena, puis d'utiliser SerDe bibliothèques JSON.

Prenons, par exemple, une application IoT qui publie des événements avec des champs communs de différents capteurs. L'un de ces champs doit stocker une charge utile personnalisée qui est propre au capteur qui envoie l'événement. Dans ce cas, comme vous ne connaissez pas le schéma, nous vous recommandons de stocker les informations sous la forme d'une chaîne codée en JSON. Pour ce faire, convertissez des données de votre table Athena en JSON, comme dans l'exemple suivant. Vous pouvez également convertir les données codées en JSON en types de données Athena.

- Conversion de types de données Athena en JSON
- Conversion de JSON en types de données Athena

Conversion de types de données Athena en JSON

Pour convertir des données Athena en types de données JSON, utilisez `CAST`.

```
WITH dataset AS (
  SELECT
    CAST('HELLO ATHENA' AS JSON) AS hello_msg,
    CAST(12345 AS JSON) AS some_int,
    CAST(MAP(ARRAY['a', 'b'], ARRAY[1,2]) AS JSON) AS some_map
)
SELECT * FROM dataset
```

Cette requête renvoie :

```
+-----------------------------------------+
| hello_msg      | some_int | some_map      |
+-----------------------------------------+
```

```
| "HELLO ATHENA" | 12345    | {"a":1,"b":2} |
+-----------------------------------------+
```

Conversion de JSON en types de données Athena

Pour convertir des données JSON en types de données Athena, utilisez `CAST`.

> **Note**
>
> Dans cet exemple, pour indiquer que les chaînes sont codées en JSON, commencez avec le mot-clé `JSON` et utilisez des apostrophes, par exemple, `JSON '12345'`

```
WITH dataset AS (
  SELECT
    CAST(JSON '"HELLO ATHENA"' AS VARCHAR) AS hello_msg,
    CAST(JSON '12345' AS INTEGER) AS some_int,
    CAST(JSON '{"a":1,"b":2}' AS MAP(VARCHAR, INTEGER)) AS some_map
)
SELECT * FROM dataset
```

Cette requête renvoie :

```
+-------------------------------------+
| hello_msg    | some_int | some_map  |
+-------------------------------------+
| HELLO ATHENA | 12345    | {a:1,b:2} |
+-------------------------------------+
```

Extraction de données JSON à partir de chaînes

Vous pouvez disposer de données source contenant des chaînes codées en JSON que vous ne souhaitez pas forcément désérialiser dans une table dans Athena. Dans ce cas, vous pouvez toujours exécuter des opérations SQL sur ces données à l'aide des fonctions JSON disponibles dans Presto.

Envisagez cette chaîne JSON comme exemple d'ensemble de données.

```
{"name": "Susan Smith",
"org": "engineering",
```

```
"projects":
    [
     {"name":"project1", "completed":false},
     {"name":"project2", "completed":true}
    ]
}
```

Exemples : extraction de propriétés

Pour extraire les propriétés `name` et `projects` de la chaîne JSON, utilisez la fonction `json_extract` comme dans l'exemple suivant. La fonction `json_extract` utilise la colonne contenant la chaîne JSON et effectue une recherche dans celle-ci à l'aide d'une expression de type `JSONPath` avec la notation de points `.`.

> **Note**
>
> `JSONPath` effectue un parcours simple de l'arborescence. Cette expression utilise le signe $ pour indiquer la racine du document JSON, suivi d'un point et d'un élément imbriqué directement sous la racine, par exemple `$.name`.

```
WITH dataset AS (
  SELECT '{"name": "Susan Smith",
           "org": "engineering",
           "projects": [{"name":"project1", "completed":false},
           {"name":"project2", "completed":true}]}'
    AS myblob
)
SELECT
  json_extract(myblob, '$.name') AS name,
  json_extract(myblob, '$.projects') AS projects
FROM dataset
```

La valeur renvoyée est une chaîne codée en JSON et non un type de données Athena natif.

```
+-------------------------------------------------------------------------------------------
+
| name           | projects
         |
+-------------------------------------------------------------------------------------------
+
```

```
| "Susan Smith"  | [{"name":"project1","completed":false},
{"name":"project2","completed":true}] |
+-----------------------------------------------------------------------------------------------------
+
```

Pour extraire la valeur scalaire de la chaîne JSON, utilisez la fonction `json_extract_scalar`. Celle-ci est similaire à `json_extract`, mais elle renvoie uniquement des valeurs scalaires (booléennes, numériques ou de type chaîne).

> **Note**
>
> N'utilisez pas la fonction `json_extract_scalar` sur des tableaux, des mappages ou des structures.

```
WITH dataset AS (
  SELECT '{"name": "Susan Smith",
           "org": "engineering",
           "projects": [{"name":"project1", "completed":false},{"name":"project2",
 "completed":true}]}'
    AS myblob
)
SELECT
  json_extract_scalar(myblob, '$.name') AS name,
  json_extract_scalar(myblob, '$.projects') AS projects
FROM dataset
```

Cette requête renvoie :

```
+---------------------------+
| name           | projects |
+---------------------------+
| Susan Smith    |          |
+---------------------------+
```

Pour obtenir le premier élément de la propriété `projects` dans l'exemple de tableau, utilisez la fonction `json_array_get` et spécifiez la position d'index.

```
WITH dataset AS (
  SELECT '{"name": "Bob Smith",
           "org": "engineering",
```

```
           "projects": [{"name":"project1", "completed":false},{"name":"project2",
 "completed":true}]}'
    AS myblob
)
SELECT json_array_get(json_extract(myblob, '$.projects'), 0) AS item
FROM dataset
```

Cette fonction renvoie la valeur à la position d'index spécifiée dans le tableau codé en JSON.

```
+--------------------------------------+
| item                                 |
+--------------------------------------+
| {"name":"project1","completed":false} |
+--------------------------------------+
```

Pour renvoyer un type de chaîne Athena, utilisez l'opérateur `[]` à l'intérieur d'une expression `JSONPath`, puis utilisez la fonction `json_extract_scalar`. Pour plus d'informations sur `[]`, consultez Accès à des éléments de tableau.

```
WITH dataset AS (
   SELECT '{"name": "Bob Smith",
             "org": "engineering",
             "projects": [{"name":"project1", "completed":false},{"name":"project2",
 "completed":true}]}'
     AS myblob
)
SELECT json_extract_scalar(myblob, '$.projects[0].name') AS project_name
FROM dataset
```

Elle renvoie le résultat suivant :

```
+--------------+
| project_name |
+--------------+
| project1     |
+--------------+
```

Recherche de valeurs dans les tableaux JSON

Afin de déterminer si une valeur spécifique existe dans un tableau codé au format JSON, utilisez la fonction `json_array_contains`.

La requête suivante répertorie les noms des utilisateurs qui participent à « project2 ».

```
WITH dataset AS (
  SELECT * FROM (VALUES
    (JSON '{"name": "Bob Smith", "org": "legal", "projects": ["project1"]}'),
    (JSON '{"name": "Susan Smith", "org": "engineering", "projects": ["project1",
 "project2", "project3"]}'),
    (JSON '{"name": "Jane Smith", "org": "finance", "projects": ["project1",
 "project2"]}')
  ) AS t (users)
)
SELECT json_extract_scalar(users, '$.name') AS user
FROM dataset
WHERE json_array_contains(json_extract(users, '$.projects'), 'project2')
```

Cette requête renvoie une liste d'utilisateurs.

```
+-------------+
| user        |
+-------------+
| Susan Smith |
+-------------+
| Jane Smith  |
+-------------+
```

L'exemple de requête suivant répertorie les noms des utilisateurs qui ont terminé des projets, ainsi que le nombre total de projets terminés. La requête effectue les actions suivantes :

- Utilise les instructions imbriquées `SELECT` pour plus de clarté.
- Extrait le tableau de projets.
- Convertit le tableau en un tableau natif de paires clé-valeur utilisant `CAST`.
- Extrait chaque élément individuel du tableau à l'aide de l'opérateur `UNNEST`.
- Filtre les valeurs obtenues en fonction des projets terminés et en fournit un décompte.

> **Note**
>
> Lorsque vous utilisez CAST pour MAP, vous pouvez spécifier VARCHAR en tant qu'élément clé (chaîne native dans Presto), mais laisser JSON pour la valeur, car les valeurs de MAP sont de types différents : chaîne pour la première paire clé-valeur, et booléen pour la seconde.

```
WITH dataset AS (
  SELECT * FROM (VALUES
    (JSON '{"name": "Bob Smith",
             "org": "legal",
             "projects": [{"name":"project1", "completed":false}]}'),
    (JSON '{"name": "Susan Smith",
             "org": "engineering",
             "projects": [{"name":"project2", "completed":true},
                          {"name":"project3", "completed":true}]}'),
    (JSON '{"name": "Jane Smith",
             "org": "finance",
             "projects": [{"name":"project2", "completed":true}]}')
  ) AS t (users)
),
employees AS (
  SELECT users, CAST(json_extract(users, '$.projects') AS
    ARRAY(MAP(VARCHAR, JSON))) AS projects_array
  FROM dataset
),
names AS (
  SELECT json_extract_scalar(users, '$.name') AS name, projects
  FROM employees, UNNEST (projects_array) AS t(projects)
)
SELECT name, count(projects) AS completed_projects FROM names
WHERE cast(element_at(projects, 'completed') AS BOOLEAN) = true
GROUP BY name
```

Cette requête renvoie le résultat suivant :

```
+---------------------------------+
| name        | completed_projects |
+---------------------------------+
| Susan Smith | 2                  |
+---------------------------------+
| Jane Smith  | 1                  |
```

```
+---------------------------------+
```

Obtention de la longueur et de la taille de tableaux JSON

Exemple : **json_array_length**

Pour obtenir la longueur d'un tableau codé en JSON, utilisez la fonction `json_array_length`.

```
WITH dataset AS (
  SELECT * FROM (VALUES
    (JSON '{"name":
            "Bob Smith",
            "org":
            "legal",
            "projects": [{"name":"project1", "completed":false}]}'),
    (JSON '{"name": "Susan Smith",
            "org": "engineering",
            "projects": [{"name":"project2", "completed":true},
                         {"name":"project3", "completed":true}]}'),
    (JSON '{"name": "Jane Smith",
             "org": "finance",
             "projects": [{"name":"project2", "completed":true}]}')
  ) AS t (users)
)
SELECT
  json_extract_scalar(users, '$.name') as name,
  json_array_length(json_extract(users, '$.projects')) as count
FROM dataset
ORDER BY count DESC
```

Cette requête renvoie le résultat suivant :

```
+---------------------+
| name        | count |
+---------------------+
| Susan Smith | 2     |
+---------------------+
| Bob Smith   | 1     |
+---------------------+
| Jane Smith  | 1     |
+---------------------+
```

Exemple : `json_size`

Pour obtenir la taille d'un tableau ou d'un objet codé en JSON, utilisez la fonction `json_size`, puis spécifiez la colonne contenant la chaîne JSON et l'expression `JSONPath` sur le tableau ou l'objet.

```
WITH dataset AS (
  SELECT * FROM (VALUES
    (JSON '{"name": "Bob Smith", "org": "legal", "projects": [{"name":"project1",
 "completed":false}]}'),
    (JSON '{"name": "Susan Smith", "org": "engineering", "projects":
 [{"name":"project2", "completed":true},{"name":"project3", "completed":true}]}'),
    (JSON '{"name": "Jane Smith", "org": "finance", "projects": [{"name":"project2",
 "completed":true}]}')
  ) AS t (users)
)
SELECT
  json_extract_scalar(users, '$.name') as name,
  json_size(users, '$.projects') as count
FROM dataset
ORDER BY count DESC
```

Cette requête renvoie le résultat suivant :

```
+----------------------+
| name        | count  |
+----------------------+
| Susan Smith | 2      |
+----------------------+
| Bob Smith   | 1      |
+----------------------+
| Jane Smith  | 1      |
+----------------------+
```

Résolution des problèmes de requêtes JSON

Pour obtenir de l'aide sur la résolution des problèmes liés aux requêtes JSON, conzultez Erreurs liées à JSON ou les ressources suivantes :

- J'obtiens des erreurs lorsque j'essaie de lire des données JSON dans Amazon Athena.
- Comment résoudre le problème « HIVE_CURSOR_ERROR : Row is not a valid JSON object - JSONException : Duplicate key » lors de la lecture de fichiers depuis Athena ? AWS Config

- La requête SELECT COUNT dans Amazon Athena renvoie un seul enregistrement alors que le fichier JSON d'entrée contient plusieurs enregistrements.
- Comment puis-je voir le fichier source Simple Storage Service (Amazon S3) pour une ligne dans un tableau Athena?

Voir aussi Considérations et limitations relatives aux requêtes SQL dans Amazon Athena.

Utilisation du Machine Learning (ML) avec Amazon Athena

Le Machine Learning (ML) avec Amazon Athena vous permet d'utiliser Athena pour écrire des instructions SQL qui exécutent l'inférence du Machine Learning (ML) à l'aide d'Amazon. SageMaker Cette fonctionnalité simplifie l'accès aux modèles ML pour l'analyse des données, éliminant ainsi la nécessité d'utiliser des méthodes de programmation complexes pour exécuter l'inférence.

Pour utiliser ML avec Athena, vous définissez une fonction ML avec Athena avec la clause `USING EXTERNAL FUNCTION`. La fonction pointe vers le point de terminaison du SageMaker modèle que vous souhaitez utiliser et spécifie les noms des variables et les types de données à transmettre au modèle. Les clauses suivantes de la requête font référence à la fonction de transmission des valeurs au modèle. Le modèle exécute l'inférence en fonction des valeurs que la requête transmet, puis renvoie les résultats de l'inférence. Pour plus d'informations sur les SageMaker endpoints SageMaker et leur fonctionnement, consultez le manuel Amazon SageMaker Developer Guide.

Pour un exemple d'utilisation du machine learning avec Athena et de l' SageMaker inférence pour détecter une valeur anormale dans un ensemble de résultats, consultez l'article du blog AWS Big Data sur la détection de valeurs anormales en invoquant la fonction d'inférence d'apprentissage automatique Amazon Athena.

Considérations et restrictions

- Régions disponibles — La fonctionnalité Athena ML est une fonctionnalité dans laquelle le moteur Régions AWS Athena version 2 ou ultérieure est pris en charge.
- SageMaker le point de terminaison du modèle doit accepter et renvoyer **`text/csv`** — Pour plus d'informations sur les formats de données, consultez la section Formats de données courants pour l'inférence dans le manuel Amazon SageMaker Developer Guide.
- Athena n'envoie pas d'en-têtes CSV — Si votre SageMaker point de terminaison l'est`text/csv`, votre gestionnaire de saisie ne doit pas supposer que la première ligne de l'entrée est un en-tête CSV. Comme Athena n'envoie pas d'en-têtes CSV, la sortie renvoyée à Athena contiendra une ligne de moins que ce qu'Athena attend et générera une erreur.

- SageMaker dimensionnement du point de terminaison : assurez-vous que le point de terminaison du SageMaker modèle référencé est suffisamment étendu pour les appels d'Athena au point de terminaison. Pour plus d'informations, consultez la section Mise à l'échelle automatique SageMaker des modèles dans le manuel Amazon SageMaker Developer Guide et CreateEndpointConfigdans le Amazon SageMaker API Reference.
- Autorisations IAM — Pour exécuter une requête qui spécifie un ML avec la fonction Athena, le principal IAM qui exécute la requête doit être autorisé à effectuer l'action pour le point de terminaison `sagemaker:InvokeEndpoint` du modèle référencé. SageMaker Pour plus d'informations, consultez Autorisation d'accès pour ML avec Athena.
- Les fonctions ML avec Athena ne peuvent pas être utilisées directement dans les clauses **`GROUP BY`**

ML avec syntaxe Athena

La clause `USING EXTERNAL FUNCTION` spécifie une fonction ML avec Athena ou plusieurs fonctions qui peuvent être référencées par une instruction `SELECT` ultérieure dans la requête. Vous définissez le nom de la fonction, les noms de variables et les types de données des variables et des valeurs de retour.

Résumé

La syntaxe suivante montre une `USING EXTERNAL FUNCTION` clause qui spécifie une fonction ML avec Athena.

```
USING EXTERNAL FUNCTION ml_function_name (variable1 data_type[, variable2 data_type]
[,...])
RETURNS data_type
SAGEMAKER 'sagemaker_endpoint'
SELECT ml_function_name()
```

Paramètres

UTILISATION DE LA FONCTION EXTERNE ***`ml_function_nom`*** (***`variable1 data_type`***[, ***`variable2 data_type`***][,...])

`ml_function_nom` définit le nom de la fonction, qui peut être utilisé dans les clauses de requête suivantes. Chaque *`variable data_type`* spécifie une variable nommée et le type de données correspondant que le SageMaker modèle accepte en entrée. Le type de données spécifié doit être un type de données Athena pris en charge.

RETURNS ***type_données***

data_type indique le type de données SQL que *ml_function_name* renvoie à la requête en sortie du modèle. SageMaker

SAGEMAKER '***sagemaker_endpoint***'

sagemaker_endpoint indique le point final du modèle. SageMaker

SELECT [...] ***ml_function_name***(***expression***) [...]

La requête SELECT qui transmet des valeurs aux variables de fonction et au SageMaker modèle pour renvoyer un résultat. *ml_function_name* indique la fonction définie précédemment dans la requête, suivie d'une *expression* qui est évaluée pour transmettre des valeurs. Les valeurs transmises et renvoyées doivent correspondre aux types de données correspondants spécifiés pour la fonction dans la clause `USING EXTERNAL FUNCTION`.

Exemple

L'exemple suivant illustre une requête utilisant ML avec Athena.

Example

```
USING EXTERNAL FUNCTION predict_customer_registration(age INTEGER)
    RETURNS DOUBLE
    SAGEMAKER 'xgboost-2019-09-20-04-49-29-303'
SELECT predict_customer_registration(age) AS probability_of_enrolling, customer_id
     FROM "sampledb"."ml_test_dataset"
     WHERE predict_customer_registration(age) < 0.5;
```

Exemples d'utilisation par les clients

Les vidéos suivantes, qui utilisent la version préliminaire de Machine Learning (ML) avec Amazon Athena, montrent comment vous pouvez utiliser SageMaker Athena.

Prédiction du taux de désabonnement des clients

La vidéo suivante montre comment associer Athena aux capacités d'apprentissage automatique d'Amazon SageMaker pour prévoir le taux de désabonnement des clients.

Prédisez le taux de désabonnement des clients à l'aide d'Amazon Athena et Amazon SageMaker

Détection des réseaux d'ordinateurs zombies

La vidéo suivante montre comment une entreprise utilise Amazon Athena et Amazon SageMaker pour détecter les botnets.

Détectez les botnets à l'aide d'Amazon Athena et Amazon SageMaker

Interrogation avec des fonctions définies par l'utilisateur

Les fonctions définies par l'utilisateur (UDF) dans Amazon Athena vous permettent de créer des fonctions personnalisées pour traiter des registres ou des groupes de registres. Une UDF accepte les paramètres, effectue le travail, puis renvoie un résultat.

Pour utiliser une UDF dans Athena, vous écrivez une clause `USING EXTERNAL FUNCTION` avant une instruction `SELECT` dans une requête SQL. L'instruction `SELECT` fait référence à l'UDF et définit les variables qui sont transmises à l'UDF lors de l'exécution de la requête. La requête SQL appelle une fonction Lambda en utilisant le moteur d'exécution Java lorsqu'elle invoque l'UDF. Les UDF sont définies dans la fonction Lambda en tant que méthodes dans un package de déploiement Java. Plusieurs UDF peuvent être définies dans le même package de déploiement Java pour une fonction Lambda. Vous spécifiez également le nom de la fonction Lambda dans la clause `USING EXTERNAL FUNCTION`.

Vous disposez de deux options pour déployer une fonction Lambda pour les UDF Athena. Vous pouvez déployer la fonction directement en utilisant Lambda, ou vous pouvez utiliser le AWS Serverless Application Repository. Pour trouver les fonctions Lambda existantes pour les UDF, vous pouvez effectuer une recherche dans le référentiel public AWS Serverless Application Repository ou privé, puis les déployer sur Lambda. Vous pouvez également créer ou modifier le code source Java, le packager dans un fichier JAR et le déployer à l'aide de Lambda ou du AWS Serverless Application Repository. Pour des exemples de code source Java et de packages pour vous aider à démarrer, voir Création et déploiement d'une UDF à l'aide de Lambda. Pour plus d'informations sur Lambda, consultez le Guide du développeur AWS Lambda. Pour plus d'informations AWS Serverless Application Repository, consultez le guide du AWS Serverless Application Repository développeur.

Pour un exemple d'utilisation des UDF avec Athena pour traduire et analyser du texte, consultez l'article du AWS blog Machine Learning Translate and analysis text using SQL functions with Amazon Athena, Amazon Translate et Amazon Comprehend, ou regardez le. video

Pour un exemple d'utilisation des UDF pour étendre les requêtes géospatiales dans Amazon Athena, consultez Étendre les requêtes géospatiales dans Amazon Athena avec des UDF et AWS Lambda sur le blog AWS Big Data.

Considérations et restrictions

- Fonctions Athena intégrées – Les fonctions intégrées dans Athena sont conçues pour être très performantes. Nous vous recommandons d'utiliser des fonctions intégrées plutôt que les UDF lorsque cela est possible. Pour de plus amples informations sur les fonctions intégrées, veuillez consulter Fonctions dans Amazon Athena.
- UDF scalaires uniquement – Athena ne prend en charge que les UDF scalaires, qui traitent une ligne à la fois et renvoient une valeur de colonne unique. Athena transmet un lot de lignes, potentiellement en parallèle, à l'UDF chaque fois qu'il invoque Lambda. Lors de la conception des UDF et des requêtes, tenez compte de l'impact potentiel de ce traitement sur le trafic réseau.
- Les fonctions du gestionnaire de UDF utilisent un format abrégé — Utilisez un format abrégé (pas le format complet), pour vos fonctions UDF (par exemple, `package.Class` et non de `package.Class::method`).
- Les méthodes UDF doivent être en minuscules — Les méthodes UDF doivent être en minuscules ; la casse chameau n'est pas autorisée.
- Les méthodes UDF nécessitent des paramètres – Les méthodes UDF doivent avoir au moins un paramètre d'entrée. Toute tentative d'invocation d'une méthode UDF définie sans paramètres d'entrée entraîne une exception d'exécution. Les UDF sont destinés à exécuter des fonctions sur des enregistrements de données, mais un UDF sans arguments n'accepte aucune donnée, générant une exception.
- Prise en charge de l'exécution Java – Actuellement, les UDF Athena prennent en charge les exécutions Java 8 et Java 11 pour Lambda. Pour plus d'informations, consultez la rubrique Création de fonctions Lambda avec Java du Guide du développeur AWS Lambda .
- Autorisations IAM – Pour exécuter et créer des instructions de requête UDF dans Athena, le principal IAM qui exécute la requête doit être autorisé à effectuer des actions en plus des fonctions Athena. Pour plus d'informations, consultez Exemple de politiques d'autorisations IAM pour autoriser les fonctions définies par l'utilisateur (UDF) Amazon Athena.
- Quotas Lambda – Les quotas Lambda s'appliquent aux UDF. Pour plus d'informations, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .
- Filtrage au niveau des lignes – Le filtrage au niveau des lignes de Lake Formation n'est pas pris en charge pour les UDF.
- Views (Vues) : vous ne pouvez pas utiliser des vues avec des UDF.
- Problèmes connus — Pour obtenir la up-to-date liste la plus complète des problèmes connus, voir Limitations et problèmes dans la section awslabs/ aws-athena-query-federation de. GitHub

Vidéos

Regardez les vidéos suivantes pour en savoir plus sur l'utilisation des fonctions UDF dans Athena.

Vidéo : Présentation des fonctions définies par l'utilisateur (UDF) dans Amazon Athena

La vidéo suivante montre comment utiliser les fichiers UDF dans Amazon Athena pour effacer des informations sensibles.

> **Note**
>
> La syntaxe de cette vidéo est une version préliminaire, mais les concepts sont les mêmes. Utilisez Athena sans le groupe de travail `AmazonAthenaPreviewFunctionality`.

[Présentation des fonctions définies par l'utilisateur (UDF) dans Amazon Athena]

Vidéo : Traduction, analyse et rédaction de champs de texte à l'aide de requêtes SQL dans Amazon Athena

La vidéo suivante montre comment vous pouvez utiliser les UDF dans Amazon Athena avec d'autres Services AWS pour traduire et analyser du texte.

> **Note**
>
> La syntaxe de cette vidéo est une version préliminaire, mais les concepts sont les mêmes. Pour connaître la syntaxe correcte, consultez l'article de blog intitulé Traduction, rédaction et analyse de texte à l'aide de fonctions SQL avec Amazon Athena, Amazon Translate et Amazon Comprehend sur le blog AWS Machine Learning.

[Traduction, analyse et rédaction de champs de texte à l'aide de requêtes SQL dans Amazon Athena]

Syntaxe de requête UDF

La clause `USING EXTERNAL FUNCTION` spécifie une ou plusieurs UDF qui peuvent être référencées par une instruction `SELECT` ultérieure dans la requête. Vous avez besoin du nom de la méthode de l'UDF et du nom de la fonction Lambda qui héberge l'UDF. À la place du nom de la fonction Lambda, vous pouvez utiliser l'ARN Lambda. Dans les scénarios entre comptes, l'ARN Lambda est requis.

Résumé

```
USING EXTERNAL FUNCTION UDF_name(variable1 data_type[, variable2 data_type][,...])
RETURNS data_type
LAMBDA 'lambda_function_name_or_ARN'
[, EXTERNAL FUNCTION UDF_name2(variable1 data_type[, variable2 data_type][,...])
RETURNS data_type
LAMBDA 'lambda_function_name_or_ARN'[,...]]
SELECT  [...] UDF_name(expression) [, UDF_name2(expression)] [...]
```

Paramètres

USING EXTERNAL FUNCTION ***UDF_name***(***variable1 data_type***[, ***variable2 data_type***][,...])

UDF_name spécifie le nom de l'UDF, qui doit correspondre à une méthode Java au sein de la fonction Lambda référencée. Chaque élément *data_type variable* spécifie une variable nommée et son type de données correspondant que l'UDF accepte en entrée. Le *data_type* doit être l'un des types de données Athena pris en charge, répertoriés dans le tableau suivant, et être mappé sur le type de données Java correspondant.

Type de données Athena	Type de données Java
TIMESTAMP	java.time. LocalDateTime (UTC)
DATE	java.time. LocalDate (UTC)
TINYINT	java.lang.Byte
SMALLINT	java.lang.Short
REAL	java.lang.Float
DOUBLE	java.lang.Double
DECIMAL (voir la note RETURNS)	java.math. BigDecimal
BIGINT	java.lang.Long

Type de données Athena	Type de données Java
INTEGER	java.lang.Int
VARCHAR	java.lang.String
VARBINARY	byte[]
BOOLEAN	java.lang.Boolean
ARRAY	java.util.List
ROW	java.util.Map<String, Object>

RETURNS ***type_données***

`data_type` spécifie le type de données SQL que l'UDF renvoie en sortie. Les types de données Athena répertoriés dans le tableau ci-dessus sont pris en charge. Pour le type de données `DECIMAL`, utilisez la syntaxe `RETURNS DECIMAL(`*`precision`*`, `*`scale`*`)` où la *`précision`* et l'*`échelle`* sont des entiers.

LAMBDA '***lambda_function***'

`lambda_function` spécifie le nom de la fonction Lambda à invoquer lors de l'exécution de l'UDF.

SELECT [...] ***UDF_name***(***expression***) [...]

La requête `SELECT` qui transmet des valeurs à l'UDF et renvoie un résultat. *`UDF_name`* spécifie l'UDF à utiliser, suivie d'une *`expression`* qui est évaluée pour transmettre des valeurs. Les valeurs transmises et renvoyées doivent correspondre aux types de données correspondants spécifiés pour l'UDF dans la clause `USING EXTERNAL FUNCTION`.

Exemples

Pour des exemples de requêtes basées sur le code AthenaUDFHandler.java activé GitHub, consultez la page du GitHub connecteur Amazon Athena UDF.

Création et déploiement d'une UDF à l'aide de Lambda

Pour créer une UDF personnalisée, vous créez une nouvelle classe Java en étendant la classe `UserDefinedFunctionHandler`. Le code source du UserDefinedFunctionHandlerfichier .java dans le SDK est disponible GitHub dans le athena-federation-sdk référentiel awslabs/aws-athena-query-federation/, ainsi que des exemples d'implémentations UDF que vous pouvez examiner et modifier pour créer un UDF personnalisé.

Les étapes de cette section illustrent l'écriture et la création d'un fichier Jar UDF personnalisé à l'aide d' Apache Maven à partir de la ligne de commande, ainsi qu'un déploiement.

Étapes de création d'une UDF personnalisée pour Athena à l'aide de Maven

- Cloner le kit SDK et préparer votre environnement de développement
- Créez votre projet Maven
- Ajouter des dépendances et des plugins à votre projet Maven
- Écrire du code Java pour les UDF
- Construire le fichier JAR
- Déployez le JAR sur AWS Lambda

Cloner le kit SDK et préparer votre environnement de développement

Avant de commencer, assurez-vous que git est installé sur votre système en utilisant `sudo yum install git -y`.

Pour installer le SDK de fédération de AWS requêtes

- Saisissez ce qui suit sur la ligne de commande pour cloner le référentiel SDK. Ce référentiel inclut le kit SDK, des exemples et une suite de connecteurs de source de données. Pour de plus amples informations sur les connecteurs de source de données, veuillez consulter Utilisation de la requête fédérée d'Amazon Athena.

  ```
  git clone https://github.com/awslabs/aws-athena-query-federation.git
  ```

Pour installer les éléments requis pour cette procédure

Si vous travaillez sur une machine de développement sur laquelle Apache Maven, le AWS CLI, et l'outil de AWS Serverless Application Model compilation sont déjà installés, vous pouvez ignorer cette étape.

1. À partir de la racine du répertoire `aws-athena-query-federation` que vous avez créé lors du clonage, exécutez le script prepare_dev_env.sh qui prépare votre environnement de développement.
2. Mettez à jour votre shell pour obtenir les nouvelles variables créées par le processus d'installation ou redémarrez votre session de terminal.

```
source ~/.profile
```

> **Important**
>
> Si vous ignorez cette étape, vous recevrez ultérieurement des erreurs indiquant que l'outil AWS CLI ou l'outil de AWS SAM génération ne sera pas en mesure de publier votre fonction Lambda.

Créez votre projet Maven

Exécutez la commande suivante pour créer votre projet Maven. Remplacez *GroupID* par l'identifiant unique de votre organisation et remplacez-le *my-athena-udf*par le nom de votre application. Pour plus d'informations, consultez Comment créer mon premier projet Maven ? dans la documentation d'Apache Maven.

```
mvn -B archetype:generate \
-DarchetypeGroupId=org.apache.maven.archetypes \
-DgroupId=groupId \
-DartifactId=my-athena-udfs
```

Ajouter des dépendances et des plugins à votre projet Maven

Ajoutez les configurations suivantes au fichier `pom.xml` du projet Maven. Pour un exemple, consultez le fichier pom.xml dans GitHub.

```
<properties>
```

```
    <aws-athena-federation-sdk.version>2022.47.1</aws-athena-federation-sdk.version>
</properties>

<dependencies>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-athena-federation-sdk</artifactId>
        <version>${aws-athena-federation-sdk.version}</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>3.2.1</version>
            <configuration>
                <createDependencyReducedPom>false</createDependencyReducedPom>
                <filters>
                    <filter>
                        <artifact>*:*</artifact>
                        <excludes>
                            <exclude>META-INF/*.SF</exclude>
                            <exclude>META-INF/*.DSA</exclude>
                            <exclude>META-INF/*.RSA</exclude>
                        </excludes>
                    </filter>
                </filters>
            </configuration>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>shade</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Écrire du code Java pour les UDF

Créez une nouvelle classe en étendant le UserDefinedFunctionHandlerfichier .java. Écrivez vos UDF à l'intérieur de la classe.

Dans l'exemple suivant, deux méthodes Java pour les UDF, `compress()` et `decompress()`, sont créées à l'intérieur de la classe `MyUserDefinedFunctions`.

```
*package *com.mycompany.athena.udfs;

public class MyUserDefinedFunctions
        extends UserDefinedFunctionHandler
{
    private static final String SOURCE_TYPE = "MyCompany";

    public MyUserDefinedFunctions()
    {
        super(SOURCE_TYPE);
    }

    /**
     * Compresses a valid UTF-8 String using the zlib compression library.
     * Encodes bytes with Base64 encoding scheme.
     *
     * @param input the String to be compressed
     * @return the compressed String
     */
    public String compress(String input)
    {
        byte[] inputBytes = input.getBytes(StandardCharsets.UTF_8);

        // create compressor
        Deflater compressor = new Deflater();
        compressor.setInput(inputBytes);
        compressor.finish();

        // compress bytes to output stream
        byte[] buffer = new byte[4096];
        ByteArrayOutputStream byteArrayOutputStream = new
 ByteArrayOutputStream(inputBytes.length);
        while (!compressor.finished()) {
            int bytes = compressor.deflate(buffer);
            byteArrayOutputStream.write(buffer, 0, bytes);
```

```
        }

        try {
            byteArrayOutputStream.close();
        }
        catch (IOException e) {
            throw new RuntimeException("Failed to close ByteArrayOutputStream", e);
        }

        // return encoded string
        byte[] compressedBytes = byteArrayOutputStream.toByteArray();
        return Base64.getEncoder().encodeToString(compressedBytes);
    }

    /**
     * Decompresses a valid String that has been compressed using the zlib compression
library.
     * Decodes bytes with Base64 decoding scheme.
     *
     * @param input the String to be decompressed
     * @return the decompressed String
     */
    public String decompress(String input)
    {
        byte[] inputBytes = Base64.getDecoder().decode((input));

        // create decompressor
        Inflater decompressor = new Inflater();
        decompressor.setInput(inputBytes, 0, inputBytes.length);

        // decompress bytes to output stream
        byte[] buffer = new byte[4096];
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(inputBytes.length);
        try {
            while (!decompressor.finished()) {
                int bytes = decompressor.inflate(buffer);
                if (bytes == 0 && decompressor.needsInput()) {
                    throw new DataFormatException("Input is truncated");
                }
                byteArrayOutputStream.write(buffer, 0, bytes);
            }
        }
        catch (DataFormatException e) {
```

```
            throw new RuntimeException("Failed to decompress string", e);
        }

        try {
            byteArrayOutputStream.close();
        }
        catch (IOException e) {
            throw new RuntimeException("Failed to close ByteArrayOutputStream", e);
        }

        // return decoded string
        byte[] decompressedBytes = byteArrayOutputStream.toByteArray();
        return new String(decompressedBytes, StandardCharsets.UTF_8);
    }
}
```

Construire le fichier JAR

Exécutez `mvn clean install` pour construire votre projet. Une fois la construction terminée, un fichier JAR est créé dans le dossier `target` de votre projet nommé *`artifactId`*`-`*`version`*`.jar`, où *`artifactId`* est le nom que vous avez fourni dans le projet Maven, par exemple, `my-athena-udfs`.

Déployez le JAR sur AWS Lambda

Vous avez deux options pour déployer votre code sur Lambda :

- Déployer à l'aide de AWS Serverless Application Repository (recommandé)
- Création d'une fonction Lambda à partir du fichier JAR

Option 1 : déploiement vers le AWS Serverless Application Repository

Lorsque vous déployez votre fichier JAR sur le AWS Serverless Application Repository, vous créez un AWS SAM modèle de fichier YAML qui représente l'architecture de votre application. Vous spécifiez ensuite ce fichier YAML et un compartiment Simple Storage Service (Amazon S3) où les artefacts de votre application sont téléchargés et mis à disposition du AWS Serverless Application Repository. La procédure ci-dessous utilise le script publish.sh situé dans le répertoire `athena-query-federation/tools` du kit Athena Query Federation SDK que vous avez cloné précédemment.

Pour plus d'informations et pour connaître les exigences, consultez les sections Publication d'applications dans le Guide du AWS Serverless Application Repository développeur, concepts de AWS SAM modèles dans le Guide du AWS Serverless Application Model développeur et Publication d'applications sans serveur à l'aide de la AWS SAM CLI.

L'exemple suivant illustre les paramètres d'un fichier YAML. Ajoutez des paramètres similaires à votre fichier YAML et enregistrez-le dans votre répertoire de projet. Voir athena-udf.yaml pour un exemple complet. GitHub

```
Transform: 'AWS::Serverless-2016-10-31'
Metadata:
  'AWS::ServerlessRepo::Application':
    Name: MyApplicationName
    Description: 'The description I write for my application'
    Author: 'Author Name'
    Labels:
      - athena-federation
    SemanticVersion: 1.0.0
Parameters:
  LambdaFunctionName:
    Description: 'The name of the Lambda function that will contain your UDFs.'
    Type: String
  LambdaTimeout:
    Description: 'Maximum Lambda invocation runtime in seconds. (min 1 - 900 max)'
    Default: 900
    Type: Number
  LambdaMemory:
    Description: 'Lambda memory in MB (min 128 - 3008 max).'
    Default: 3008
    Type: Number
Resources:
  ConnectorConfig:
    Type: 'AWS::Serverless::Function'
    Properties:
      FunctionName: !Ref LambdaFunctionName
      Handler: "full.path.to.your.handler. For example,
 com.amazonaws.athena.connectors.udfs.MyUDFHandler"
      CodeUri: "Relative path to your JAR file. For example, ./target/athena-
udfs-1.0.jar"
      Description: "My description of the UDFs that this Lambda function enables."
      Runtime: java8
      Timeout: !Ref LambdaTimeout
```

```
      MemorySize: !Ref LambdaMemory
```

Copiez le script `publish.sh` dans le répertoire du projet dans lequel vous avez enregistré votre fichier YAML et exécutez la commande suivante :

```
./publish.sh MyS3Location MyYamlFile
```

Par exemple, si l'emplacement de votre compartiment est `s3://DOC-EXAMPLE-BUCKET/mysarapps/athenaudf` et que votre fichier YAML a été enregistré sous la forme `my-athena-udfs.yaml` :

```
./publish.sh DOC-EXAMPLE-BUCKET/mysarapps/athenaudf my-athena-udfs
```

Pour créer une fonction Lambda

1. Ouvrez la console Lambda à l'adresse https://console.aws.amazon.com/lambda/, choisissez Create function (Créer une fonction), puis Browse serverless app repository (Parcourir le référentiel d'applications sans serveur)
2. Choisissez Private applications (Applications privées), recherchez votre application dans la liste ou recherchez-la en utilisant des mots clés, puis sélectionnez-la.
3. Vérifiez et fournissez les détails de l'application, puis choisissez Déployer.

   Vous pouvez maintenant utiliser les noms de méthode définis dans le fichier JAR de votre fonction Lambda comme des UDF dans Athena.

Option 2 : création directe d'une fonction Lambda

Vous pouvez également créer une fonction Lambda directement à l'aide de la console ou. AWS CLI L'exemple suivant illustre l'utilisation de la commande CLI `create-function` Lambda.

```
aws lambda create-function \
 --function-name MyLambdaFunctionName \
 --runtime java8 \
 --role arn:aws:iam::1234567890123:role/my_lambda_role \
 --handler com.mycompany.athena.udfs.MyUserDefinedFunctions \
 --timeout 900 \
 --zip-file fileb://./target/my-athena-udfs-1.0-SNAPSHOT.jar
```

Requête entre régions

Athena permet d'interroger les données Amazon S3 dans une région différente de Région AWS celle dans laquelle vous utilisez Athena. La requête entre régions peut être une option lorsque le déplacement des données n'est pas pratique ou autorisé, ou si vous souhaitez interroger des données dans plusieurs régions. Même si Athena n'est pas disponible dans une région particulière, les données de cette région peuvent être interrogées à partir d'une autre région dans laquelle Athena est disponible.

Pour interroger des données dans une région, votre compte doit être activé dans cette région même si les données Simple Storage Service (Amazon S3) n'appartiennent pas à votre compte. Pour certaines régions telles que USA Est (Ohio), votre accès à la région est automatiquement activé lorsque votre compte est créé. Les autres régions exigent que votre compte soit « inscrit » à la région avant de pouvoir l'utiliser. Pour obtenir la liste des régions nécessitant un opt-in, consultez la section Régions disponibles dans le guide de l'utilisateur Amazon EC2. Pour obtenir des instructions spécifiques sur l'adhésion à une région, consultez la section Gestion des AWS régions dans le. Référence générale d'Amazon Web Services

Considérations et restrictions

- Autorisations d'accès de données – Pour pouvoir interroger avec succès les données Simple Storage Service (Amazon S3) depuis Athena dans toutes les régions, votre compte doit disposer des autorisations nécessaires pour lire les données. Si les données que vous souhaitez interroger appartiennent à un autre compte, l'autre compte doit vous accorder l'accès à l'emplacement Simple Storage Service (Amazon S3) qui contient les données.
- Frais de transfert de données : des frais de transfert de données Amazon S3 peuvent s'appliquer aux requêtes entre régions. L'exécution d'une requête peut entraîner le transfert d'un volume de données supérieur à la taille du jeu de données. Nous vous recommandons de commencer par tester vos requêtes sur un sous-jeu de données et d'en examiner les coûts dans AWS Cost Explorer.
- AWS Glue— Vous pouvez l'utiliser AWS Glue dans toutes les régions. Des frais supplémentaires peuvent s'appliquer pour le AWS Glue trafic interrégional. Pour plus d'informations, consultez la section Créer des AWS Glue connexions entre comptes et entre régions dans le blog AWS Big Data.
- Options de chiffrement Simple Storage Service (Amazon S3)— Les options de chiffrement SSE-S3 et SSE-KMS sont prises en charge pour les requêtes dans toutes les régions ; CSE-KMS

ne l'est pas. Pour plus d'informations, consultez Options de chiffrement Simple Storage Service (Amazon S3) prises en charge.

- Requêtes fédérées : l'utilisation de requêtes fédérées entre elles n' Régions AWS est pas prise en charge.
- Régions chinoises — Les requêtes interrégionales ne sont pas prises en charge dans les régions chinoises.

Si les conditions ci-dessus sont remplies, vous pouvez créer une table Athena pointant vers la valeur `LOCATION` que vous spécifiez et interroger les données de manière transparente. Aucune syntaxe spéciale n'est requise. Pour plus d'informations sur la création de tables Athena, veuillez consulter Création de tables dans Athena.

Interrogation du AWS Glue Data Catalog

Étant donné que le AWS Glue Data Catalog est utilisé par de nombreux Services AWS comme référentiel central de métadonnées, vous voulez peut-être interroger les métadonnées du catalogue de données. Pour ce faire, vous pouvez utiliser des requêtes SQL dans Athena. Vous pouvez utiliser Athena pour interroger des métadonnées du catalogue AWS Glue telles que des bases de données, des tables, des partitions et des colonnes.

Pour obtenir les métadonnées du catalogue AWS Glue, vous interrogez la base de données `information_schema` sur le backend Athena. Les exemples de requête de cette rubrique montrent comment utiliser Athena pour interroger les métadonnées du catalogue AWS Glue pour les cas d'utilisation courants.

Rubriques

Considérations et restrictions

- Au lieu de faire d'interroger la base de données `information_schema`, il est possible d'utiliser des commandes DDL Apache Hive individuelles pour extraire des informations sur les métadonnées pour des bases de données, des tables, des vues, des partitions et des colonnes spécifiques à partir d'Athena. Toutefois, le résultat est dans un format non tabulaire.
- Les requêtes à `information_schema` sont plus performantes si vous avez une quantité faible à modérée de métadonnées AWS Glue. Si vous disposez d'un grand nombre de métadonnées, des erreurs peuvent se produire.
- Vous ne pouvez pas utiliser `CREATE VIEW` pour créer une vue sur la base de données `information_schema`.

Liste des bases de données et recherche dans une base de données spécifiée

Les exemples de cette section montrent comment répertorier les bases de données dans les métadonnées par nom de schéma.

Example – Liste des bases de données

L'exemple de requête suivant répertorie les bases de données de la table `information_schema.schemata`.

```
SELECT schema_name
FROM   information_schema.schemata
LIMIT  10;
```

Le tableau suivant présente des exemples de résultats.

6	alb-databas1
7	alb_original_cust
8	alblogsdatabase
9	athena_db_test
10	athena_ddl_db

Example – Recherche dans une base de données spécifiée

Dans l'exemple de requête suivant, `rdspostgresql` est un exemple de base de données.

```
SELECT schema_name
FROM   information_schema.schemata
WHERE  schema_name = 'rdspostgresql'
```

Le tableau suivant présente des exemples de résultats.

	nom_schéma
1	rdspostgresql

Liste des tables dans une base de données spécifiée et recherche d'une table par nom

Pour répertorier les métadonnées des tables, vous pouvez effectuer une interrogation par schéma de table ou par nom de table.

Example – Liste des tables par schéma

La requête suivante répertorie les tables qui utilisent le schéma de table `rdspostgresql`.

```
SELECT table_schema,
       table_name,
       table_type
FROM   information_schema.tables
WHERE  table_schema = 'rdspostgresql'
```

Le tableau suivant montre un exemple de résultat.

	table_schema	table_name	table_type
1	rdspostgresql	rdspostgresqldb1_public_account	BASE TABLE

Example – Recherche d'une table par nom

La requête suivante obtient des informations de métadonnées pour la table `athena1`.

```
SELECT table_schema,
       table_name,
       table_type
FROM   information_schema.tables
WHERE  table_name = 'athena1'
```

Le tableau suivant montre un exemple de résultat.

	table_schema	table_name	table_type
1	default	athena1	BASE TABLE

Liste des partitions d'une table spécifique

Vous pouvez utiliser `SHOW PARTITIONS` *`table_name`* pour répertorier les partitions d'une table spécifiée, comme dans l'exemple suivant.

```
SHOW PARTITIONS cloudtrail_logs_test2
```

Vous pouvez également utiliser une requête de métadonnées `$partitions` pour répertorier les numéros de partition et les valeurs de partition d'une table spécifique.

Example – Interrogation des partitions d'une table à l'aide de la syntaxe $partitions

L'exemple de requête suivant répertorie les partitions pour la table `cloudtrail_logs_test2` en utilisant la syntaxe `$partitions`.

```
SELECT * FROM default."cloudtrail_logs_test2$partitions" ORDER BY partition_number
```

Le tableau suivant présente des exemples de résultats.

	table_catalog	table_schema	table_name	Année	Mois	jour
1	awsdatacatalog	par défaut	cloudtrail_logs_test2	2020	08	10

	table_catalog	table_schema	table_name	Année	Mois	jour
2	awsdatacatalog	par défaut	cloudtrail_logs_test2	2020	08	11
3	awsdatacatalog	par défaut	cloudtrail_logs_test2	2020	08	12

Liste de toutes les colonnes pour toutes les tables

Vous pouvez répertorier toutes les colonnes de toutes les tables dans `AwsDataCatalog` ou pour toutes les tables d'une base de données spécifique dans `AwsDataCatalog`.

- Pour répertorier toutes les colonnes de toutes les bases de données dans `AwsDataCatalog`, utilisez la requête `SELECT * FROM information_schema.columns`.
- Pour limiter les résultats à une base de données spécifique, utilisez `table_schema='database_name'` dans la clause `WHERE`.

Example – Liste de toutes les tables d'une base de données spécifique

L'exemple de requête suivant répertorie toutes les colonnes pour toutes les tables dans la base de données `webdata`.

```
SELECT * FROM information_schema.columns WHERE table_schema = 'webdata'
```

Affichage des colonnes communes à des tables spécifiques

Vous pouvez afficher les colonnes que des tables spécifiques d'une base de données ont en commun.

- Utilisez la syntaxe `SELECT column_name FROM information_schema.columns`.
- Pour la clause `WHERE`, utilisez la syntaxe `WHERE table_name IN ('table1', 'table2')`.

Example – Affichage des colonnes communes à deux tables de la même base de données

L'exemple de requête suivant affiche les colonnes que les tables `table1` et `table2` ont en commun.

```
SELECT column_name
FROM information_schema.columns
WHERE table_name IN ('table1', 'table2')
GROUP BY column_name
HAVING COUNT(*) > 1;
```

Liste ou recherche de colonnes pour une table ou une vue spécifiée

Vous pouvez répertorier toutes les colonnes d'une table, toutes les colonnes d'une vue ou rechercher une colonne par nom dans une base de données et une table spécifiées.

Pour répertorier les colonnes, utilisez une requête `SELECT *`. Dans la clause `FROM`, spécifiez `information_schema.columns`. Dans la clause `WHERE`, utilisez `table_schema='database_name'` pour spécifier la base de données et `table_name = 'table_name'` pour spécifier la table ou la vue qui a les colonnes que vous voulez répertorier.

Example – Liste de toutes les colonnes d'une table spécifiée

L'exemple de requête suivant répertorie toutes les colonnes de la table `rdspostgresqldb1_public_account`.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'rdspostgresql'
       AND table_name = 'rdspostgresqldb1_public_account'
```

Le tableau suivant présente des exemples de résultats.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	comm	extra_info
1	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	passwor	1		OUI	varcha		
2	awsdatacatalog	rdspostgresql	rdspostgresqldb1_p	user_id	2		OUI	entier		

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	comm	extra_info
			ublic_account							
3	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	created_n	3		OUI	timest		
4	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	last_login	4		OUI	timest		
5	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	Amazon	5		OUI	varcha		
6	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	usernam	6		OUI	varcha		

Example – Liste des colonnes d'une vue spécifiée

L'exemple de requête suivant répertorie toutes les colonnes de la base de données `default` de la vue `arrayview`.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'default'
       AND table_name = 'arrayview'
```

Le tableau suivant présente des exemples de résultats.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_type	comment	extra_info
1	awsdatacatalog	par défaut	arrayview	searchdate	1		OUI	varchar		
2	awsdatacatalog	par défaut	arrayview	sid	2		OUI	varchar		
3	awsdatacatalog	par défaut	arrayview	btid	3		OUI	varchar		
4	awsdatacatalog	par défaut	arrayview	p	4		OUI	varchar		
5	awsdatacatalog	par défaut	arrayview	infantprice	5		OUI	varchar		
6	awsdatacatalog	par défaut	arrayview	sump	6		OUI	varchar		
7	awsdatacatalog	par défaut	arrayview	journeymaparray	7		OUI	array(varchar)		

Example – Recherche d'une colonne par nom dans une base de données et une table spécifiées

L'exemple de requête suivant recherche les métadonnées de la colonne `sid` dans la vue `arrayview` de la base de données `default`.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'default'
       AND table_name = 'arrayview'
       AND column_name='sid'
```

Le tableau suivant montre un exemple de résultat.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	comm	extra_info
1	awsdatacatalog	par défaut	arrayview	sid	2		OUI	varcha		

Journaux d'interrogation Service AWS

Cette section inclut plusieurs procédures d'utilisation d'Amazon Athena pour interroger des ensembles de données courants, tels que les journaux, les CloudFront journaux Amazon AWS CloudTrail , les journaux Classic Load Balancer, les journaux Application Load Balancer, les journaux de flux Amazon VPC et les journaux Network Load Balancer.

Les tâches de cette section utilisent la console Athena, mais vous pouvez également utiliser d'autres outils comme le pilote Athena JDBC, la AWS CLI ou la Référence API Amazon Athena.

Pour plus d'informations sur la AWS CloudFormation création automatique de tables de Service AWS journaux, de partitions et d'exemples de requêtes dans Athena, consultez la section Automatisation de la création de tables de Service AWS journaux et interrogation de celles-ci avec Amazon Athena sur le blog Big Data. AWS Pour plus d'informations sur l'utilisation d'une bibliothèque Python AWS Glue afin de créer une structure commune pour traiter les Service AWS journaux et les interroger dans Athena, consultez la section Interrogation aisée des journaux à l'aide d' Service AWS Amazon Athena.

Les rubriques de cette section supposent que vous avez configuré les autorisations appropriées pour accéder à Athena et au compartiment Amazon S3 où doivent résider les données à interroger. Pour plus d'informations, consultez Configuration et Mise en route.

Rubriques

- Interrogation des journaux de l'Application Load Balancer
- Interrogation des journaux de Classic Load Balancer
- Interrogation des journaux Amazon CloudFront
- Journaux d'interrogation AWS CloudTrail
- Interrogation des journaux Amazon EMR
- Interrogation des journaux AWS Global Accelerator de flux

- Interroger les résultats d'Amazon GuardDuty
- Journaux d'interrogation AWS Network Firewall
- Interrogation des journaux du dispositif du Network Load Balancer
- Interrogation des journaux de requête d'Amazon Route 53 Resolver
- Interrogation des journaux d'événements Amazon SES
- Interrogation des journaux de flux Amazon VPC
- Journaux d'interrogation AWS WAF

Interrogation des journaux de l'Application Load Balancer

Application Load Balancer est une option de répartition de charge pour Elastic Load Balancing qui permet la distribution du trafic dans un déploiement de microservices à l'aide de conteneurs. L'interrogation des journaux de l'Application Load Balancer vous permet de connaître la source du trafic et la latence, ainsi que les octets transférés vers et depuis les instances Elastic Load Balancing et les applications backend. Pour plus d'informations, consultez les journaux d'accès de votre Application Load Balancer et les journaux de connexion de votre Application Load Balancer dans le Guide de l'utilisateur des Application Load Balancers.

Rubriques

- Prérequis
- Création de la table pour les journaux d'accès ALB
- Création de la table pour les journaux d'accès ALB dans Athena à l'aide de la projection de partitions
- Exemples de requêtes pour les journaux d'accès ALB
- Création de la table pour les journaux de connexion ALB
- Création de la table pour les journaux de connexion ALB dans Athena à l'aide de la projection de partitions
- Exemples de requêtes pour les journaux de connexion ALB
- Ressources supplémentaires

Prérequis

- Activez la journalisation des accès ou la journalisation des connexions afin que les journaux Application Load Balancer puissent être enregistrés dans votre compartiment Amazon S3.

- Une base de données pour contenir la table que vous allez créer pour Athena. Pour créer une base de données, vous pouvez utiliser l'Athena ou AWS Glue la console. Pour de plus amples informations, veuillez consulter Création de bases de données dans Athena dans ce guide ou Utilisation des bases de données sur la console AWS glue dans le Guide du développeur AWS Glue .

Création de la table pour les journaux d'accès ALB

1. Copiez et collez l'`CREATE TABLE`instruction suivante dans l'éditeur de requêtes de la console Athena, puis modifiez-la selon vos propres exigences en matière de saisie dans le journal. Pour plus d'informations sur le démarrage avec la console Athena, veuillez consulter la rubrique Mise en route. Remplacez le chemin indiqué dans la `LOCATION` clause par l'emplacement de votre dossier de journal d'accès Amazon S3. Pour plus d'informations sur l'emplacement des fichiers journaux d'accès, consultez la section Fichiers journaux d'accès dans le Guide de l'utilisateur des équilibreurs de charge d'application.

   Pour plus d'informations sur chaque champ du fichier journal, consultez les entrées du journal d'accès dans le guide de l'utilisateur pour les équilibreurs de charge d'application.

   > **Note**
   >
   > L'exemple d'`CREATE TABLE`instruction suivant inclut les colonnes`classification`,`classification_reason`, et `conn_trace_id` (« ID de traçabilité », ou TID) récemment ajoutées. Pour créer une table pour les journaux d'accès à Application Load Balancer qui ne contiennent pas ces entrées, supprimez les colonnes correspondantes de l'`CREATE TABLE`instruction et modifiez l'expression régulière en conséquence.

```
CREATE EXTERNAL TABLE IF NOT EXISTS alb_access_logs (
             type string,
             time string,
             elb string,
             client_ip string,
             client_port int,
             target_ip string,
             target_port int,
             request_processing_time double,
```

```
            target_processing_time double,
            response_processing_time double,
            elb_status_code int,
            target_status_code string,
            received_bytes bigint,
            sent_bytes bigint,
            request_verb string,
            request_url string,
            request_proto string,
            user_agent string,
            ssl_cipher string,
            ssl_protocol string,
            target_group_arn string,
            trace_id string,
            domain_name string,
            chosen_cert_arn string,
            matched_rule_priority string,
            request_creation_time string,
            actions_executed string,
            redirect_url string,
            lambda_error_reason string,
            target_port_list string,
            target_status_code_list string,
            classification string,
            classification_reason string,
            conn_trace_id string
            )
            ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
            WITH SERDEPROPERTIES (
            'serialization.format' = '1',
            'input.regex' =
        '([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*)[:-]([0-9]*) ([-.0-9]*)
 ([-.0-9]*) ([-.0-9]*) (|[-0-9]*) (-|[-0-9]*) ([-0-9]*) ([-0-9]*) \"([^ ]*) (.*) (-
 |[^ ]*)\" \"([^\"]*)\" ([A-Z0-9-_]+) ([A-Za-z0-9.-]*) ([^ ]*) \"([^\"]*)\" \"([^
\"]*)\" \"([^\"]*)\" ([-.0-9]*) ([^ ]*) \"([^\"]*)\" \"([^\"]*)\" \"([^ ]*)\" \"([^
\s]+?)\" \"([^\s]+)\" \"([^ ]*)\" \"([^ ]*)\" ?([^ ]*)?( .*)?')
            LOCATION 's3://DOC-EXAMPLE-BUCKET/access-log-folder-path/'
```

2. Exécutez la requête dans la console Athena. Une fois que la requête est terminée, Athena enregistre la table `alb_access_logs`, de telle sorte que les données soient prêtes pour que vous puissiez émettre des requêtes.

Création de la table pour les journaux d'accès ALB dans Athena à l'aide de la projection de partitions

Les journaux d'accès ALB ayant une structure connue dont vous pouvez spécifier le schéma de partition à l'avance, vous pouvez réduire le temps d'exécution des requêtes et automatiser la gestion des partitions en utilisant la fonction de projection de partition Athena. La projection des partitions ajoute automatiquement de nouvelles partitions à mesure que de nouvelles données sont ajoutées. Vous n'avez donc plus besoin d'ajouter manuellement des partitions à l'aide de la commande `ALTER TABLE ADD PARTITION`.

L'exemple d'`CREATE TABLE`instruction suivant utilise automatiquement la projection de partition sur les journaux d'accès ALB à partir d'une date spécifiée jusqu'à aujourd'hui pour une seule AWS région. L'instruction se base sur l'exemple de la section précédente, mais ajoute les clauses `PARTITIONED BY` et `TBLPROPERTIES` pour activer la projection de partition. Dans les `storage.location.template` clauses `LOCATION` et, remplacez les espaces réservés par des valeurs identifiant l'emplacement du compartiment Amazon S3 de vos journaux d'accès ALB. Pour plus d'informations sur l'emplacement des fichiers journaux d'accès, consultez la section Fichiers journaux d'accès dans le Guide de l'utilisateur des équilibreurs de charge d'application. Pour `projection.day.range`, remplacez *`2022`*/*`01`*/*`01`* par la date de début que vous souhaitez utiliser. Après avoir exécuté la requête avec succès, vous pouvez interroger la table. Vous n'avez pas besoin d'exécuter `ALTER TABLE ADD PARTITION` pour charger les partitions. Pour plus d'informations sur chaque champ du fichier journal, consultez la section Entrées du journal d'accès.

```
CREATE EXTERNAL TABLE IF NOT EXISTS alb_access_logs (
             type string,
             time string,
             elb string,
             client_ip string,
             client_port int,
             target_ip string,
             target_port int,
             request_processing_time double,
             target_processing_time double,
             response_processing_time double,
             elb_status_code int,
             target_status_code string,
             received_bytes bigint,
             sent_bytes bigint,
             request_verb string,
             request_url string,
             request_proto string,
             user_agent string,
```

```
            ssl_cipher string,
            ssl_protocol string,
            target_group_arn string,
            trace_id string,
            domain_name string,
            chosen_cert_arn string,
            matched_rule_priority string,
            request_creation_time string,
            actions_executed string,
            redirect_url string,
            lambda_error_reason string,
            target_port_list string,
            target_status_code_list string,
            classification string,
            classification_reason string,
            conn_trace_id string
            )
            PARTITIONED BY
            (
             day STRING
            )
            ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
            WITH SERDEPROPERTIES (
            'serialization.format' = '1',
            'input.regex' =
        '([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*)[:-]([0-9]*) ([-.0-9]*)
 ([-.0-9]*) ([-.0-9]*) ([-.0-9]*) (|[-0-9]*) (-|[-0-9]*) ([-0-9]*) ([-0-9]*) \"([^ ]*) (.*) (- |
[^ ]*)\" \"([^\"]*)\" ([A-Z0-9-_]+) ([A-Za-z0-9.-]*) ([^ ]*) \"([^\"]*)\" \"([^\"]*)\"
 \"([^\"]*)\" ([-.0-9]*) ([^ ]*) \"([^\"]*)\" \"([^\"]*)\" \"([^ ]*)\" \"([^\s]+?)\"
 \"([^\s]+)\" \"([^ ]*)\" \"([^ ]*)\" ?([^ ]*)?( .*)?')
            LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/<ACCOUNT-NUMBER>/
elasticloadbalancing/<REGION>/'
            TBLPROPERTIES
            (
             "projection.enabled" = "true",
             "projection.day.type" = "date",
             "projection.day.range" = "2022/01/01,NOW",
             "projection.day.format" = "yyyy/MM/dd",
             "projection.day.interval" = "1",
             "projection.day.interval.unit" = "DAYS",
             "storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/AWSLogs/<ACCOUNT-
NUMBER>/elasticloadbalancing/<REGION>/${day}"
            )
```

Pour plus d'informations sur la projection de partition, voir Projection de partition avec Amazon Athena.

Exemples de requêtes pour les journaux d'accès ALB

La requête suivante compte le nombre de demandes HTTP GET reçues par l'équilibreur de charge et regroupées par l'adresse IP du client:

```
SELECT COUNT(request_verb) AS
 count,
 request_verb,
 client_ip
FROM alb_access_logs
GROUP BY request_verb, client_ip
LIMIT 100;
```

Une autre requête affiche les URL visitées par les utilisateurs du navigateur Safari:

```
SELECT request_url
FROM alb_access_logs
WHERE user_agent LIKE '%Safari%'
LIMIT 10;
```

La requête suivante montre les enregistrements dont les valeurs de code d'état ELB sont supérieures ou égales à 500.

```
SELECT * FROM alb_access_logs
WHERE elb_status_code >= 500
```

L'exemple suivant montre comment analyser les journaux par `datetime` :

```
SELECT client_ip, sum(received_bytes)
FROM alb_access_logs
WHERE parse_datetime(time,'yyyy-MM-dd''T''HH:mm:ss.SSSSSS''Z')
     BETWEEN parse_datetime('2018-05-30-12:00:00','yyyy-MM-dd-HH:mm:ss')
     AND parse_datetime('2018-05-31-00:00:00','yyyy-MM-dd-HH:mm:ss')
GROUP BY client_ip;
```

La requête suivante interroge la table qui utilise la projection de partition pour tous les journaux d'accès ALB à partir du jour spécifié.

```
SELECT *
FROM alb_access_logs
WHERE day = '2022/02/12'
```

Création de la table pour les journaux de connexion ALB

1. Copiez et collez l'exemple d'`CREATE TABLE`instruction suivant dans l'éditeur de requêtes de la console Athena, puis modifiez-le selon vos propres exigences en matière de saisie dans le journal. Pour plus d'informations sur le démarrage avec la console Athena, veuillez consulter la rubrique Mise en route. Remplacez le chemin indiqué dans la `LOCATION` clause par l'emplacement du dossier de votre journal de connexion Amazon S3. Pour plus d'informations sur l'emplacement des fichiers journaux de connexion, consultez la section Fichiers journaux de connexion dans le Guide de l'utilisateur des équilibreurs de charge d'application. Pour plus d'informations sur chaque champ du fichier journal, consultez la section Entrées du journal des connexions.

   ```
   CREATE EXTERNAL TABLE IF NOT EXISTS alb_connection_logs (
            time string,
            client_ip string,
            client_port int,
            listener_port int,
            tls_protocol string,
            tls_cipher string,
            tls_handshake_latency double,
            leaf_client_cert_subject string,
            leaf_client_cert_validity string,
            leaf_client_cert_serial_number string,
            tls_verify_status string,
            conn_trace_id string
            )
            ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
            WITH SERDEPROPERTIES (
            'serialization.format' = '1',
            'input.regex' =
             '([^ ]*) ([^ ]*) ([0-9]*) ([0-9]*) ([A-Za-z0-9.-]*) ([^ ]*) ([-.0-9]*)
    \"([^\"]*)\" ([^ ]*) ([^ ]*) ([^ ]*) ?([^ ]*)?( .*)?')
            LOCATION 's3://DOC-EXAMPLE-BUCKET/connection-log-folder-path/'
   ```

2. Exécutez la requête dans la console Athena. Une fois que la requête est terminée, Athena enregistre la table `alb_connection_logs`, de telle sorte que les données soient prêtes pour que vous puissiez émettre des requêtes.

Création de la table pour les journaux de connexion ALB dans Athena à l'aide de la projection de partitions

Comme les journaux de connexion ALB ont une structure connue dont vous pouvez spécifier le schéma de partition à l'avance, vous pouvez réduire le temps d'exécution des requêtes et automatiser la gestion des partitions en utilisant la fonction de projection de partition Athena. La projection des partitions ajoute automatiquement de nouvelles partitions à mesure que de nouvelles données sont ajoutées. Vous n'avez donc plus besoin d'ajouter manuellement des partitions à l'aide de la commande `ALTER TABLE ADD PARTITION`.

L'exemple d'`CREATE TABLE`instruction suivant utilise automatiquement la projection de partition sur les journaux de connexion ALB à partir d'une date spécifiée jusqu'à aujourd'hui pour une seule AWS région. L'instruction se base sur l'exemple de la section précédente, mais ajoute les clauses `PARTITIONED BY` et `TBLPROPERTIES` pour activer la projection de partition. Dans les `storage.location.template` clauses `LOCATION` et, remplacez les espaces réservés par des valeurs identifiant l'emplacement du compartiment Amazon S3 de vos journaux de connexion ALB. Pour plus d'informations sur l'emplacement des fichiers journaux de connexion, consultez la section Fichiers journaux de connexion dans le Guide de l'utilisateur des équilibreurs de charge d'application. Pour`projection.day.range`, remplacez *2023*/*01*/*01* par la date de début que vous souhaitez utiliser. Après avoir exécuté la requête avec succès, vous pouvez interroger la table. Vous n'avez pas besoin d'exécuter `ALTER TABLE ADD PARTITION` pour charger les partitions. Pour plus d'informations sur chaque champ du fichier journal, consultez la section Entrées du journal des connexions.

```
CREATE EXTERNAL TABLE IF NOT EXISTS alb_connection_logs (
         time string,
         client_ip string,
         client_port int,
         listener_port int,
         tls_protocol string,
         tls_cipher string,
         tls_handshake_latency double,
         leaf_client_cert_subject string,
         leaf_client_cert_validity string,
         leaf_client_cert_serial_number string,
         tls_verify_status string,
         conn_trace_id string
         )
            PARTITIONED BY
            (
```

```
             day STRING
            )
            ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
            WITH SERDEPROPERTIES (
            'serialization.format' = '1',
            'input.regex' =
             '([^ ]*) ([^ ]*) ([0-9]*) ([0-9]*) ([A-Za-z0-9.-]*) ([^ ]*) ([-.0-9]*)
 \"([^\"]*)\" ([^ ]*) ([^ ]*) ([^ ]*) ?([^ ]*)?( .*)?')
            LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/<ACCOUNT-NUMBER>/
elasticloadbalancing/<REGION>/'
            TBLPROPERTIES
            (
             "projection.enabled" = "true",
             "projection.day.type" = "date",
             "projection.day.range" = "2023/01/01,NOW",
             "projection.day.format" = "yyyy/MM/dd",
             "projection.day.interval" = "1",
             "projection.day.interval.unit" = "DAYS",
             "storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/AWSLogs/<ACCOUNT-
NUMBER>/elasticloadbalancing/<REGION>/${day}"
            )
```

Pour plus d'informations sur la projection de partition, voir Projection de partition avec Amazon Athena.

Exemples de requêtes pour les journaux de connexion ALB

Les requêtes suivantes comptent les occurrences pour lesquelles la valeur de n'`tls_verify_status`était pas`'Success'`, groupées par adresse IP du client :

```
SELECT DISTINCT client_ip, count() AS count FROM alb_connection_logs
WHERE tls_verify_status != 'Success'
GROUP BY client_ip
ORDER BY count() DESC;
```

La requête suivante recherche les occurrences pour lesquelles la valeur de `tls_handshake_latency` était supérieure à 2 secondes dans la plage de temps spécifiée :

```
SELECT * FROM alb_connection_logs
WHERE
  (
    parse_datetime(time, 'yyyy-MM-dd''T''HH:mm:ss.SSSSSS''Z')
    BETWEEN
```

```
    parse_datetime('2024-01-01-00:00:00', 'yyyy-MM-dd-HH:mm:ss')
    AND
    parse_datetime('2024-03-20-00:00:00', 'yyyy-MM-dd-HH:mm:ss')
  )
  AND
    (tls_handshake_latency >= 2.0);
```

Ressources supplémentaires

- Comment analyser mes journaux d'accès Application Load Balancer avec Amazon Athena ? dans le Centre de connaissances AWS .
- Pour plus d'informations sur les codes d'état HTTP dans Elastic Load Balancing, consultez la section Résolution des problèmes de vos Application Load Balancers dans le Guide de l'utilisateur des Application Load Balancers.
- Cataloguez et analysez les journaux d'Application Load Balancer de manière plus efficace grâce à des classificateurs AWS Glue personnalisés et à Amazon Athena sur le Big Data Blog.AWS

Interrogation des journaux de Classic Load Balancer

Utilisez les journaux Classic Load Balancer pour analyser et comprendre les modèles de trafic en direction et en provenance des instances Elastic Load Balancing et des applications backend. Vous pouvez voir la source du trafic, la latence et les octets qui ont été transférés.

Avant d'analyser les journaux Elastic Load Balancing, configurez-les pour les enregistrer dans le compartiment Simple Storage Service (Amazon S3) de destination. Pour plus d'informations, consultez Activation des journaux d'accès pour votre Classic Load Balancer.

- Création de la table pour les journaux Elastic Load Balancing
- Exemple de requêtes Elastic Load Balancing

Créer la table pour les journaux Elastic Load Balancing

1. Copiez et collez l'instruction DDL suivante dans la console Athena. Vérifiez la syntaxe des registres du journal Elastic Load Balancing. Il se peut que vous ayez besoin de mettre à jour la requête suivante afin d'inclure les colonnes et la syntaxe d'expression régulière (Regex) pour la dernière version de l'enregistrement.

   ```
   CREATE EXTERNAL TABLE IF NOT EXISTS elb_logs (
   ```

```
 timestamp string,
 elb_name string,
 request_ip string,
 request_port int,
 backend_ip string,
 backend_port int,
 request_processing_time double,
 backend_processing_time double,
 client_response_time double,
 elb_response_code string,
 backend_response_code string,
 received_bytes bigint,
 sent_bytes bigint,
 request_verb string,
 url string,
 protocol string,
 user_agent string,
 ssl_cipher string,
 ssl_protocol string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
 'serialization.format' = '1',
 'input.regex' = '([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*)[:-]([0-9]*) ([-.0-9]*)
 ([-.0-9]*) ([-.0-9]*) (|[-0-9]*) (-|[-0-9]*) ([-0-9]*) ([-0-9]*) \\\"([^ ]*)
 ([^ ]*) (- |[^ ]*)\\\" (\"[^\"]*\") ([A-Z0-9-]+) ([A-Za-z0-9.-]*)$'
)
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/AWS_account_ID/elasticloadbalancing/';
```

2. Modifiez le compartiment Simple Storage Service (Amazon S3) `LOCATION` pour spécifier la destination de vos journaux Elastic Load Balancing.
3. Exécutez la requête dans la console Athena. Une fois la requête terminée, Athena enregistre la table `elb_logs`, afin d'en préparer les données pour les requêtes. Pour plus d'informations, consultez Exemple de requêtes Elastic Load Balancing.

Exemple de requêtes Elastic Load Balancing

Utilisez une requête similaire à l'exemple suivant. Elle répertorie les serveurs d'application backend qui ont renvoyé un code de réponse d'erreur `4XX` ou `5XX`. Utilisez l'opérateur `LIMIT` pour limiter le nombre de journaux à interroger à la fois.

```
SELECT
 timestamp,
 elb_name,
 backend_ip,
 backend_response_code
FROM elb_logs
WHERE backend_response_code LIKE '4%' OR
      backend_response_code LIKE '5%'
LIMIT 100;
```

Utilisez une requête ultérieure pour résumer le temps de réponse de toutes les transactions regroupées par l'adresse IP backend et le nom d'instance Elastic Load Balancing.

```
SELECT sum(backend_processing_time) AS
 total_ms,
 elb_name,
 backend_ip
FROM elb_logs WHERE backend_ip <> ''
GROUP BY backend_ip, elb_name
LIMIT 100;
```

Pour plus d'informations, consultez Analyse de données dans S3 avec Athena.

Interrogation des journaux Amazon CloudFront

Vous pouvez configurer Amazon CloudFront CDN pour exporter les journaux d'accès à la distribution Web vers Amazon Simple Storage Service. Utilisez ces journaux pour explorer les habitudes de navigation des utilisateurs sur les propriétés Web que vous utilisez CloudFront.

Avant de commencer à interroger les journaux, activez le journal d'accès aux distributions Web sur votre CloudFront distribution préférée. Pour plus d'informations, consultez les journaux d'accès dans le manuel Amazon CloudFront Developer Guide. Notez le compartiment Amazon S3 dans lequel vous enregistrez ces journaux.

- Création d'une table pour les journaux CloudFront standard
- Création d'un tableau pour les journaux CloudFront en temps réel
- Exemples de requêtes pour les CloudFront journaux standard

Création d'une table pour les journaux CloudFront standard

> **Note**
>
> Cette procédure fonctionne pour les connexions d'accès à la distribution Web CloudFront. Elle ne s'applique pas aux journaux de streaming issus des distributions RTMP.

Pour créer une table pour les champs de fichier journal CloudFront standard

1. Copiez et collez l'exemple d'instruction DDL suivante dans l'éditeur de requête de la console Athena. L'exemple d'instruction utilise les champs du fichier journal décrits dans la section Champs de fichier journal standard du Amazon CloudFront Developer Guide. Modifiez l'emplacement (`LOCATION`) pour le compartiment Simple Storage Service (Amazon S3) qui stocke vos journaux. Pour plus d'informations sur l'utilisation de l'éditeur de requête, voir Mise en route.

   Cette requête indique `ROW FORMAT DELIMITED` et `FIELDS TERMINATED BY '\t'` indique que les champs sont délimités par des tabulations. En `ROW FORMAT DELIMITED` effet, Athéna utilise le LazySimpleSerDepar défaut. La colonne `date` est une séquence d'échappement avec des guillemets simples inversés (`), car il s'agit d'un mot réservé dans Athena. Pour plus d'informations, veuillez consulter Mots-clés réservés.

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_standard_logs (
  `date` DATE,
  time STRING,
  x_edge_location STRING,
  sc_bytes BIGINT,
  c_ip STRING,
  cs_method STRING,
  cs_host STRING,
  cs_uri_stem STRING,
  sc_status INT,
  cs_referrer STRING,
  cs_user_agent STRING,
  cs_uri_query STRING,
  cs_cookie STRING,
  x_edge_result_type STRING,
  x_edge_request_id STRING,
  x_host_header STRING,
  cs_protocol STRING,
  cs_bytes BIGINT,
```

```
  time_taken FLOAT,
  x_forwarded_for STRING,
  ssl_protocol STRING,
  ssl_cipher STRING,
  x_edge_response_result_type STRING,
  cs_protocol_version STRING,
  fle_status STRING,
  fle_encrypted_fields INT,
  c_port INT,
  time_to_first_byte FLOAT,
  x_edge_detailed_result_type STRING,
  sc_content_type STRING,
  sc_content_len BIGINT,
  sc_range_start BIGINT,
  sc_range_end BIGINT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
TBLPROPERTIES ( 'skip.header.line.count'='2' )
```

2. Exécutez la requête dans la console Athena. Une fois que la requête est terminée, Athena enregistre la table `cloudfront_standard_logs`, de telle sorte que les données soient prêtes pour que vous puissiez émettre des requêtes.

Création d'un tableau pour les journaux CloudFront en temps réel

Pour créer une table pour les champs du fichier journal CloudFront en temps réel

1. Copiez et collez l'exemple d'instruction DDL suivante dans l'éditeur de requête de la console Athena. L'exemple d'instruction utilise les champs du fichier journal décrits dans la section Journaux en temps réel du Amazon CloudFront Developer Guide. Modifiez l'emplacement (`LOCATION`) pour le compartiment Simple Storage Service (Amazon S3) qui stocke vos journaux. Pour plus d'informations sur l'utilisation de l'éditeur de requête, voir Mise en route.

   Cette requête indique `ROW FORMAT DELIMITED` et `FIELDS TERMINATED BY '\t'` indique que les champs sont délimités par des tabulations. En `ROW FORMAT DELIMITED` effet, Athéna utilise le LazySimpleSerDepar défaut. La colonne `timestamp` est une séquence d'échappement avec des guillemets simples inversés (`), car il s'agit d'un mot réservé dans Athena. Pour plus d'informations, veuillez consulter Mots-clés réservés.

L'exemple suivant contient tous les champs disponibles. Vous pouvez ajouter des commentaires ou supprimer des champs dont vous n'avez pas besoin.

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_real_time_logs (
`timestamp` STRING,
c_ip STRING,
time_to_first_byte BIGINT,
sc_status BIGINT,
sc_bytes BIGINT,
cs_method STRING,
cs_protocol STRING,
cs_host STRING,
cs_uri_stem STRING,
cs_bytes BIGINT,
x_edge_location STRING,
x_edge_request_id STRING,
x_host_header STRING,
time_taken BIGINT,
cs_protocol_version STRING,
c_ip_version STRING,
cs_user_agent STRING,
cs_referer STRING,
cs_cookie STRING,
cs_uri_query STRING,
x_edge_response_result_type STRING,
x_forwarded_for STRING,
ssl_protocol STRING,
ssl_cipher STRING,
x_edge_result_type STRING,
fle_encrypted_fields STRING,
fle_status STRING,
sc_content_type STRING,
sc_content_len BIGINT,
sc_range_start STRING,
sc_range_end STRING,
c_port BIGINT,
x_edge_detailed_result_type STRING,
c_country STRING,
cs_accept_encoding STRING,
cs_accept STRING,
cache_behavior_path_pattern STRING,
cs_headers STRING,
```

```
cs_header_names STRING,
cs_headers_count BIGINT,
primary_distribution_id STRING,
primary_distribution_dns_name STRING,
origin_fbl STRING,
origin_lbl STRING,
asn STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
TBLPROPERTIES ( 'skip.header.line.count'='2' )
```

2. Exécutez la requête dans la console Athena. Une fois que la requête est terminée, Athena enregistre la table `cloudfront_real_time_logs`, de telle sorte que les données soient prêtes pour que vous puissiez émettre des requêtes.

Exemples de requêtes pour les CloudFront journaux standard

La requête suivante additionne le nombre d'octets servis CloudFront entre le 9 juin et le 11 juin 2018. Placez le nom de colonne de date entre guillemets doubles car c'est un mot réservé.

```
SELECT SUM(bytes) AS total_bytes
FROM cloudfront_standard_logs
WHERE "date" BETWEEN DATE '2018-06-09' AND DATE '2018-06-11'
LIMIT 100;
```

Pour éliminer les lignes en double (par exemple, les lignes vides en double) des résultats de la requête, vous pouvez utiliser l'instruction `SELECT DISTINCT`, comme dans l'exemple suivant.

```
SELECT DISTINCT *
FROM cloudfront_standard_logs
LIMIT 10;
```

Ressources supplémentaires

Pour plus d'informations sur l'utilisation d'Athena pour interroger CloudFront les journaux, consultez les articles suivants du blog AWS Big Data.

Interrogez facilement Service AWS les journaux à l'aide d'Amazon Athena (29 mai 2019).

Analysez vos journaux CloudFront d'accès Amazon à grande échelle (21 décembre 2018).

Créez une architecture sans serveur pour analyser les journaux CloudFront d'accès Amazon à l'aide AWS Lambda d'Amazon Athena et d'Amazon Managed Service pour Apache Flink (26 mai 2017).

Journaux d'interrogation AWS CloudTrail

AWS CloudTrail est un service qui enregistre les appels AWS d'API et les événements pour les comptes Amazon Web Services.

CloudTrail les journaux contiennent des informations sur tous les appels d'API qui vous sont adressés Services AWS, y compris à la console. CloudTrail génère des fichiers journaux chiffrés et les stocke dans Amazon S3. Pour plus d'informations, consultez le Guide de l'utilisateur AWS CloudTrail.

> **Note**
>
> Si vous souhaitez effectuer des requêtes SQL sur des informations relatives à des CloudTrail événements concernant des comptes, des régions et des dates, pensez à utiliser CloudTrail Lake. CloudTrail Lake est une AWS alternative à la création de sentiers qui regroupe les informations d'une entreprise dans une banque de données événementielle unique et consultable. Au lieu d'utiliser le stockage par compartiment Amazon S3, il stocke les événements dans un lac de données, ce qui permet des requêtes plus riches et plus rapides. Vous pouvez l'utiliser pour créer des requêtes SQL qui recherchent des événements dans des organisations, des régions et dans des plages de temps personnalisées. Comme vous effectuez des requêtes CloudTrail Lake dans la CloudTrail console elle-même, Athena n'est pas nécessaire pour utiliser CloudTrail Lake. Pour plus d'informations, consultez la documentation de CloudTrail Lake.

L'utilisation d'Athena avec CloudTrail les journaux est un moyen efficace d'améliorer votre analyse de Service AWS l'activité. Par exemple, vous pouvez utiliser des requêtes pour identifier des tendances et isoler davantage l'activité par attributs, comme l'adresse IP source ou l'utilisateur.

Une application courante consiste à utiliser des CloudTrail journaux pour analyser l'activité opérationnelle à des fins de sécurité et de conformité. Pour plus d'informations sur un exemple détaillé, consultez le billet de blog consacré au AWS Big Data intitulé Analyser la sécurité, la conformité AWS CloudTrail et l'activité opérationnelle à l'aide d'Amazon Athena.

Vous pouvez utiliser Athena pour exécuter des requêtes sur ces fichiers journaux directement à partir de Simple Storage Service (Amazon S3), en spécifiant l'emplacement (`LOCATION`) des fichiers journaux. Vous pouvez effectuer cette opération de deux manières :

- En créant des tables pour les fichiers CloudTrail journaux directement depuis la CloudTrail console.
- En créant manuellement des tables pour les fichiers CloudTrail journaux dans la console Athena.

Rubriques

Comprendre CloudTrail les journaux et les tables Athena

Avant de commencer à créer des tables, vous devez en savoir un peu plus sur le stockage des données CloudTrail et sur leur mode de stockage. Cela peut vous aider à créer les tables dont vous avez besoin, que vous les créiez depuis la CloudTrail console ou depuis Athena.

CloudTrail enregistre les journaux sous forme de fichiers texte JSON au format gzip compressé (*.json.gzip). L'emplacement des fichiers journaux dépend de la façon dont vous configurez les sentiers, Région AWS des régions dans lesquelles vous vous connectez et d'autres facteurs.

Pour plus d'informations sur l'emplacement où les journaux sont stockés, la structure JSON et le contenu des fichiers registres, consultez les rubriques suivantes dans le AWS CloudTrail Guide de l'utilisateur :

- Trouver vos fichiers CloudTrail journaux
- CloudTrail Exemples de fichiers journaux
- CloudTrail enregistrer le contenu

- CloudTrail référence à un événement

Pour collecter des journaux et les enregistrer sur Amazon S3, activez-les CloudTrail depuis le AWS Management Console. Pour plus d'informations, consultez la rubrique Création d'un journal d'activité du Guide de l'utilisateur AWS CloudTrail .

Notez la destination du compartiment Simple Storage Service (Amazon S3) dans lequel vous enregistrez les journaux. Remplacez la `LOCATION` clause par le chemin d'accès à l'emplacement du CloudTrail journal et à l'ensemble d'objets avec lesquels vous souhaitez travailler. Cet exemple utilise une valeur `LOCATION` des journaux pour un compte particulier, mais vous pouvez utiliser le degré de spécificité qui convient à votre application.

Par exemple :

- Pour analyser des données à partir de plusieurs comptes, vous pouvez restaurer le spécificateur `LOCATION` afin d'indiquer tous les éléments `AWSLogs` en utilisant `LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/'`.
- Pour analyser les données correspondant à une date, un compte et une région spécifiques, utilisez `LOCATION 's3://DOC-EXAMPLE-BUCKET/123456789012/CloudTrail/us-east-1/2016/03/14/'`.

L'utilisation du plus haut niveau dans la hiérarchie des objets vous donne la plus grande flexibilité possible lorsque vous exécutez une requête à l'aide d'Athena.

Utilisation de la CloudTrail console pour créer une table Athena pour les journaux CloudTrail

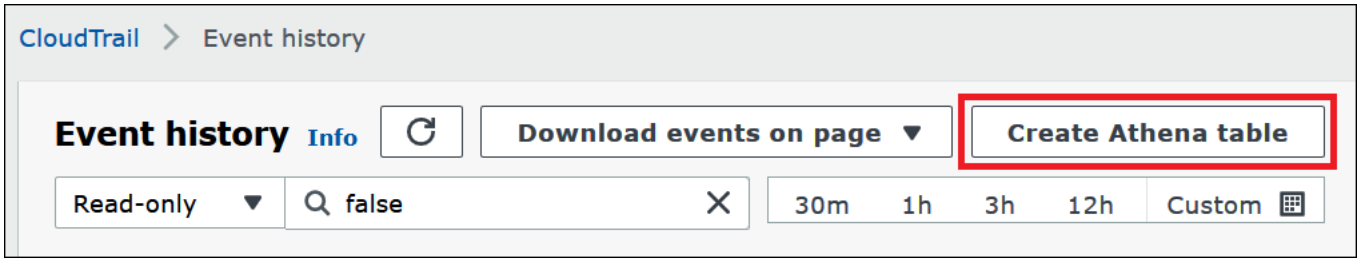
Vous pouvez créer une table Athena non partitionnée pour interroger les CloudTrail journaux directement depuis la console. CloudTrail Pour créer une table Athena depuis la CloudTrail console, vous devez être connecté avec un rôle disposant des autorisations suffisantes pour créer des tables dans Athena.

> **Note**
>
> Vous ne pouvez pas utiliser la CloudTrail console pour créer une table Athena pour les journaux de suivi des organisations. Créez plutôt la table manuellement à l'aide de la console Athena afin de pouvoir spécifier l'emplacement de stockage correct. Pour plus d'informations sur les journaux d'activité d'une organisation, consultez la rubrique Création d'un journal d'activité pour une organisation du Guide de l'utilisateur AWS CloudTrail .

- Pour plus d'informations sur la configuration des autorisations pour Athena, voir Configuration.
- Pour plus d'informations sur la création d'une table avec des partitions, consultez Création d'une table pour les CloudTrail journaux dans Athena à l'aide du partitionnement manuel.

Pour créer une table Athena pour un CloudTrail sentier à l'aide de la console CloudTrail

1. Ouvrez la CloudTrail console à l'adresse https://console.aws.amazon.com/cloudtrail/.
2. Dans le panneau de navigation, sélectionnez Event history (Historique des évènements).
3. Choisissez Create Athena table (Créer une table Athena).



4. Pour Storage location (Emplacement de stockage), utilisez la flèche vers le bas pour sélectionner le compartiment Simple Storage Service (Amazon S3) dans lequel les fichiers journaux sont stockés pour le journal d'activité à interroger.

> **Note**
>
> Pour trouver le nom du bucket associé à un parcours, choisissez Trails dans le volet de CloudTrail navigation et consultez la colonne du bucket S3 du parcours. Pour voir l'emplacement du compartiment dans Simple Storage Service (Amazon S3), choisissez le lien correspondant à ce compartiment dans la colonne Compartiment S3. Cela ouvre la console Amazon S3 à l'emplacement du CloudTrail compartiment.

5. Choisissez Créer un tableau. La table est créée avec un nom par défaut qui inclut le nom du compartiment Simple Storage Service (Amazon S3).

Création d'une table pour les CloudTrail journaux dans Athena à l'aide du partitionnement manuel

Vous pouvez créer manuellement des tables pour les fichiers CloudTrail journaux dans la console Athena, puis exécuter des requêtes dans Athena.

Pour créer une table Athena pour un CloudTrail sentier à l'aide de la console Athena

1. Copiez et collez l'instruction DDL suivante dans l'éditeur de requêtes de la console Athena.

```
CREATE EXTERNAL TABLE cloudtrail_logs (
eventversion STRING,
useridentity STRUCT<
               type:STRING,
               principalid:STRING,
               arn:STRING,
               accountid:STRING,
               invokedby:STRING,
               accesskeyid:STRING,
               userName:STRING,
  sessioncontext:STRUCT<
    attributes:STRUCT<
               mfaauthenticated:STRING,
               creationdate:STRING>,
    sessionissuer:STRUCT<
               type:STRING,
               principalId:STRING,
               arn:STRING,
               accountId:STRING,
               userName:STRING>,
    ec2RoleDelivery:string,
    webIdFederationData: STRUCT<
               federatedProvider: STRING,
               attributes: map<string,string>
    >
  >
>,
eventtime STRING,
eventsource STRING,
eventname STRING,
awsregion STRING,
sourceipaddress STRING,
useragent STRING,
errorcode STRING,
errormessage STRING,
requestparameters STRING,
responseelements STRING,
additionaleventdata STRING,
requestid STRING,
```

```
eventid STRING,
resources ARRAY<STRUCT<
               arn:STRING,
               accountid:STRING,
               type:STRING>>,
eventtype STRING,
apiversion STRING,
readonly STRING,
recipientaccountid STRING,
serviceeventdetails STRING,
sharedeventid STRING,
vpcendpointid STRING,
eventCategory STRING,
tlsDetails struct<
  tlsVersion:string,
  cipherSuite:string,
  clientProvidedHostHeader:string>
)
PARTITIONED BY (region string, year string, month string, day string)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/Account_ID/CloudTrail/';
```

> **Note**
>
> Nous vous suggérons d'utiliser ce qui `org.apache.hive.hcatalog.data.JsonSerDe` est indiqué dans l'exemple. Bien que a `com.amazon.emr.hive.serde.CloudTrailSerde` existe, il ne gère actuellement pas certains des nouveaux CloudTrail champs.

2. (Facultatif) Supprimez les champs qui ne sont pas nécessaires pour votre table. Si vous ne devez lire qu'un certain ensemble de colonnes, la définition de votre table peut exclure les autres colonnes.
3. Modifiez `s3://DOC-EXAMPLE-BUCKET/AWSLogs/Account_ID/CloudTrail/` afin qu'il pointe vers le compartiment Simple Storage Service (Amazon S3) contenant vos données de journaux.
4. Vérifiez que les champs sont répertoriés correctement. Pour plus d'informations sur la liste complète des champs d'un CloudTrail enregistrement, consultez la section Contenu de l'CloudTrail enregistrement.

L'exemple d'`CREATE TABLE` instruction de l'étape 1 utilise le Hive JSON SerDe. Dans l'exemple, les champs `requestparametersresponseelements`, et `additionaleventdata` sont répertoriés sous forme de type `STRING` dans la requête, mais sont `STRUCT` des types de données utilisés dans JSON. Ainsi, pour extraire des données de ces champs, utilisez les fonctions `JSON_EXTRACT`. Pour plus d'informations, consultez the section called "Extraction de données JSON à partir de chaînes". Pour améliorer les performances, l'exemple partitionne les données par année Région AWS, mois et jour.

5. Exécutez l'instruction `CREATE TABLE` dans la console Athena.
6. Utilisez la commande ALTER TABLE ADD PARTITION pour charger les partitions afin de pouvoir les interroger, comme dans l'exemple suivant.

```
ALTER TABLE table_name ADD
   PARTITION (region='us-east-1',
              year='2019',
              month='02',
              day='01')
   LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/Account_ID/CloudTrail/us-
east-1/2019/02/01/'
```

Création d'une table pour un journal d'activité à l'échelle de l'organisation à l'aide du partitionnement manuel

Pour créer un tableau pour les fichiers CloudTrail journaux à l'échelle de l'organisation dans Athena, suivez les étapes décrites dans la procédure suivante Création d'une table pour les CloudTrail journaux dans Athena à l'aide du partitionnement manuel, mais apportez les modifications indiquées dans la procédure suivante.

Pour créer une table Athena pour les journaux à l'échelle de l'organisation CloudTrail

1. Dans l'instruction `CREATE TABLE`, modifiez la clause `LOCATION` pour inclure l'ID de l'organisation, comme dans l'exemple suivant :

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/Account_ID/CloudTrail/'
```

2. Dans la clause `PARTITIONED BY`, ajoutez une entrée pour l'ID du compte sous forme de chaîne, comme dans l'exemple suivant :

```
PARTITIONED BY (account string, region string, year string, month string, day
 string)
```

L'exemple suivant montre le résultat combiné :

```
...

PARTITIONED BY (account string, region string, year string, month string, day
 string)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/Account_ID/CloudTrail/'
```

3. Dans l'instruction `ALTER TABLE`, clause `ADD PARTITION`, incluez l'ID du compte, comme dans l'exemple suivant :

```
ALTER TABLE table_name ADD
PARTITION (account='111122223333',
region='us-east-1',
year='2022',
month='08',
day='08')
```

4. Dans l'instruction `ALTER TABLE`, clause `LOCATION`, incluez l'ID de l'organisation, l'ID du compte et la partition que vous voulez ajouter, comme dans l'exemple suivant :

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/Account_ID/CloudTrail/us-
east-1/2022/08/08/'
```

L'exemple suivant de l'instruction `ALTER TABLE` montre le résultat combiné :

```
ALTER TABLE table_name ADD
PARTITION (account='111122223333',
region='us-east-1',
year='2022',
month='08',
day='08')
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/111122223333/CloudTrail/
us-east-1/2022/08/08/'
```

Création de la table pour les CloudTrail journaux dans Athena à l'aide de la projection de partitions

Comme CloudTrail les journaux ont une structure connue dont vous pouvez spécifier le schéma de partition à l'avance, vous pouvez réduire le temps d'exécution des requêtes et automatiser la gestion des partitions en utilisant la fonction de projection de partition Athena. La projection des partitions ajoute automatiquement de nouvelles partitions à mesure que de nouvelles données sont ajoutées. Vous n'avez donc plus besoin d'ajouter manuellement des partitions à l'aide de la commande `ALTER TABLE ADD PARTITION`.

L'exemple d'`CREATE TABLE`instruction suivant utilise automatiquement la projection de partition sur CloudTrail les journaux à partir d'une date spécifiée jusqu'à aujourd'hui pour un enregistrement individuel Région AWS. Dans les clauses `LOCATION` et `storage.location.template`, remplacez les paramètres fictifs *`bucket`*, *`account-id`* et *`aws-region`* par des valeurs identiques correspondantes. Pour `projection.timestamp.range`, remplacez *`2020`*/*`01`*/*`01`* par la date de début que vous souhaitez utiliser. Après avoir exécuté la requête avec succès, vous pouvez interroger la table. Vous n'avez pas besoin d'exécuter `ALTER TABLE ADD PARTITION` pour charger les partitions.

```
CREATE EXTERNAL TABLE cloudtrail_logs_pp(
    eventVersion STRING,
    userIdentity STRUCT<
        type: STRING,
        principalId: STRING,
        arn: STRING,
        accountId: STRING,
        invokedBy: STRING,
        accessKeyId: STRING,
        userName: STRING,
        sessionContext: STRUCT<
            attributes: STRUCT<
                mfaAuthenticated: STRING,
                creationDate: STRING>,
            sessionIssuer: STRUCT<
                type: STRING,
                principalId: STRING,
                arn: STRING,
                accountId: STRING,
                userName: STRING>,
            ec2RoleDelivery:string,
            webIdFederationData: STRUCT<
                federatedProvider: STRING,
```

```
                attributes: map<string,string>
            >
        >
    >,
    eventTime STRING,
    eventSource STRING,
    eventName STRING,
    awsRegion STRING,
    sourceIpAddress STRING,
    userAgent STRING,
    errorCode STRING,
    errorMessage STRING,
    requestparameters STRING,
    responseelements STRING,
    additionaleventdata STRING,
    requestId STRING,
    eventId STRING,
    readOnly STRING,
    resources ARRAY<STRUCT<
        arn: STRING,
        accountId: STRING,
        type: STRING>>,
    eventType STRING,
    apiVersion STRING,
    recipientAccountId STRING,
    serviceEventDetails STRING,
    sharedEventID STRING,
    vpcendpointid STRING,
    eventCategory STRING,
    tlsDetails struct<
        tlsVersion:string,
        cipherSuite:string,
        clientProvidedHostHeader:string>
  )
PARTITIONED BY (
   `timestamp` string)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/CloudTrail/aws-region'
TBLPROPERTIES (
  'projection.enabled'='true',
  'projection.timestamp.format'='yyyy/MM/dd',
```

```
  'projection.timestamp.interval'='1',
  'projection.timestamp.interval.unit'='DAYS',
  'projection.timestamp.range'='2020/01/01,NOW',
  'projection.timestamp.type'='date',
  'storage.location.template'='s3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/
CloudTrail/aws-region/${timestamp}')
```

Pour plus d'informations sur la projection de partition, voir Projection de partition avec Amazon Athena.

Interrogation des champs imbriqués

Les champs `userIdentity` et `resources` étant des types de données imbriqués, leur interrogation nécessite un traitement particulier.

L'objet `userIdentity` est constitué de types `STRUCT` imbriqués. Il est possible de les interroger en utilisant un point pour séparer les champs, comme dans l'exemple suivant :

```
SELECT
    eventsource,
    eventname,
    useridentity.sessioncontext.attributes.creationdate,
    useridentity.sessioncontext.sessionissuer.arn
FROM cloudtrail_logs
WHERE useridentity.sessioncontext.sessionissuer.arn IS NOT NULL
ORDER BY eventsource, eventname
LIMIT 10
```

Le champ `resources` est un tableau d'objets `STRUCT`. Pour ces tableaux, utilisez `CROSS JOIN UNNEST` pour désimbriquer le tableau afin de pouvoir interroger ses objets.

L'exemple suivant renvoie toutes les lignes où l'ARN de la ressource se termine par `example/datafile.txt`. Pour des raisons de lisibilité, la fonction replace (remplacer) supprime la sous-chaîne `arn:aws:s3:::` initiale de l'ARN.

```
SELECT
    awsregion,
    replace(unnested.resources_entry.ARN,'arn:aws:s3:::') as s3_resource,
    eventname,
    eventtime,
    useragent
FROM cloudtrail_logs t
```

```
CROSS JOIN UNNEST(t.resources) unnested (resources_entry)
WHERE unnested.resources_entry.ARN LIKE '%example/datafile.txt'
ORDER BY eventtime
```

L'exemple suivant interroge les événements `DeleteBucket`. La requête extrait de l'objet `resources` le nom du compartiment et l'ID du compte auquel le compartiment appartient.

```
SELECT
    awsregion,
    replace(unnested.resources_entry.ARN,'arn:aws:s3:::') as deleted_bucket,
    eventtime AS time_deleted,
    useridentity.username,
    unnested.resources_entry.accountid as bucket_acct_id
FROM cloudtrail_logs t
CROSS JOIN UNNEST(t.resources) unnested (resources_entry)
WHERE eventname = 'DeleteBucket'
ORDER BY eventtime
```

Pour plus d'informations sur la désimbrication, voir Filtrage des tableaux.

Exemple de requête

L'exemple suivant montre une partie d'une requête qui renvoie toutes les demandes anonymes (non signées) provenant de la table créée pour les journaux d' CloudTrail événements. Cette requête sélectionne ces demandes où `useridentity.accountid` est anonyme et `useridentity.arn` n'est pas spécifié :

```
SELECT *
FROM cloudtrail_logs
WHERE
    eventsource = 's3.amazonaws.com' AND
    eventname in ('GetObject') AND
    useridentity.accountid = 'anonymous' AND
    useridentity.arn IS NULL AND
    requestparameters LIKE '%[your bucket name ]%';
```

Pour plus d'informations, consultez le billet de blog consacré au AWS Big Data Analysez la sécurité, la conformité AWS CloudTrail et l'activité opérationnelle à l'aide d'Amazon Athena.

Conseils pour interroger CloudTrail les journaux

Pour explorer les données des CloudTrail journaux, suivez ces conseils :

- Avant d'exécuter des requêtes sur ces journaux, vérifiez que votre table de journaux ressemble à celle dans the section called “Création d'une table pour les CloudTrail journaux dans Athena à l'aide du partitionnement manuel”. Si ce n'est pas la première table, supprimez la table existante à l'aide de la commande suivante : `DROP TABLE cloudtrail_logs`.
- Une fois que vous la table existante supprimée recréez-la. Pour plus d’informations, consultez Création d'une table pour les CloudTrail journaux dans Athena à l'aide du partitionnement manuel.

  Vérifiez que les champs figurant dans votre requête Athena sont répertoriés correctement. Pour plus d'informations sur la liste complète des champs d'un CloudTrail enregistrement, consultez la section Contenu de l'CloudTrail enregistrement.

  Si votre requête inclut des champs dans des formats JSON, par exemple `STRUCT`, extrayez les données depuis JSON. Pour plus d’informations, consultez Extraction de données JSON à partir de chaînes.

  Quelques suggestions pour envoyer des requêtes sur votre CloudTrail table :
- Commencez par examiner quels utilisateurs ont appelé les différentes opérations d'API et à partir de quelles adresses IP sources.
- Utilisez la requête SQL de base suivante dans votre modèle. Collez la requête dans la console Athena et exécutez-la.

```
SELECT
 useridentity.arn,
 eventname,
 sourceipaddress,
 eventtime
FROM cloudtrail_logs
LIMIT 100;
```

- Modifiez la requête pour explorer davantage vos données.
- Pour améliorer les performances, insérez la clause `LIMIT` pour renvoyer un sous-ensemble spécifié de lignes.

Interrogation des journaux Amazon EMR

Amazon EMR et les applications de Big Data qui s'exécutent sur Amazon EMR produisent des fichiers journaux. Les fichiers journaux sont écrits sur le nœud principal, et vous pouvez également configurer Amazon EMR pour archiver automatiquement les fichiers journaux dans Amazon S3. Vous

pouvez utiliser Amazon Athena pour interroger ces journaux afin d'identifier les événements et les tendances pour les applications et les clusters. Pour plus d'informations sur les types de fichiers journaux dans Amazon EMR et leur enregistrement dans Simple Storage Service (Amazon S3), consultez la rubrique Affichage des fichiers journaux du Guide de gestion Amazon EMR.

Création et interrogation d'une table de base basée sur des fichiers journaux Amazon EMR

L'exemple suivant crée une table de base, `myemrlogs`, basée sur les fichiers journaux enregistrés dans `s3://aws-logs-`*`123456789012`*`-`*`us-west-2`*`/elasticmapreduce/`*`j-2ABCDE34F5GH6`*`/elasticmapreduce/`. L'emplacement Simple Storage Service (Amazon S3) utilisé dans les exemples ci-dessous reflète le modèle d'emplacement de journal par défaut pour un cluster EMR créé par le compte Amazon Web Services *`123456789012`* dans la région *`us-west-2`*. Si vous utilisez un emplacement personnalisé, le modèle est s3://DOC-EXAMPLE-BUCKET/ *`clusterID`*.

Pour plus d'informations sur la création d'une table partitionnée afin de potentiellement améliorer les performances des requêtes et réduire le transfert de données, consultez Création et interrogation d'une table partitionnée basée sur les journaux Amazon EMR.

```
CREATE EXTERNAL TABLE `myemrlogs`(
  `data` string COMMENT 'from deserializer')
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n'
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6'
```

Les exemples de requête suivants peuvent être exécutés sur la table `myemrlogs` créée par l'exemple précédent.

Example – Interrogation des journaux d'étape pour les occurrences de ERROR, WARN, INFO, EXCEPTION, FATAL ou DEBUG

```
SELECT data,
        "$PATH"
FROM "default"."myemrlogs"
WHERE regexp_like("$PATH",'s-86URH188Z6B1')
```

```
        AND regexp_like(data, 'ERROR|WARN|INFO|EXCEPTION|FATAL|DEBUG') limit 100;
```

Example – Interrogation d'un journal d'instance spécifique, i-00b3c0a839ece0a9c, pour ERROR, WARN, INFO, EXCEPTION, FATAL ou DEBUG

```
SELECT "data",
        "$PATH" AS filepath
FROM "default"."myemrlogs"
WHERE regexp_like("$PATH",'i-00b3c0a839ece0a9c')
        AND regexp_like("$PATH",'state')
        AND regexp_like(data, 'ERROR|WARN|INFO|EXCEPTION|FATAL|DEBUG') limit 100;
```

Example – Interrogation des journaux d'application Presto pour ERROR, WARN, INFO, EXCEPTION, FATAL ou DEBUG

```
SELECT "data",
        "$PATH" AS filepath
FROM "default"."myemrlogs"
WHERE regexp_like("$PATH",'presto')
        AND regexp_like(data, 'ERROR|WARN|INFO|EXCEPTION|FATAL|DEBUG') limit 100;
```

Example – Interrogation des journaux d'application Namenode pour ERROR, WARN, INFO, EXCEPTION, FATAL ou DEBUG

```
SELECT "data",
        "$PATH" AS filepath
FROM "default"."myemrlogs"
WHERE regexp_like("$PATH",'namenode')
        AND regexp_like(data, 'ERROR|WARN|INFO|EXCEPTION|FATAL|DEBUG') limit 100;
```

Example – Interrogation de tous les journaux par date et heure pour ERROR, WARN, INFO, EXCEPTION, FATAL ou DEBUG

```
SELECT distinct("$PATH") AS filepath
FROM "default"."myemrlogs"
WHERE regexp_like("$PATH",'2019-07-23-10')
        AND regexp_like(data, 'ERROR|WARN|INFO|EXCEPTION|FATAL|DEBUG') limit 100;
```

Création et interrogation d'une table partitionnée basée sur les journaux Amazon EMR

Ces exemples utilisent le même emplacement de journal pour créer une table Athena, mais la table est partitionnée et une partition est créée pour chaque emplacement de journal. Pour plus d'informations, consultez Partitionnement de données dans Athena.

La requête suivante crée la table partitionnée nommée `mypartitionedemrlogs`:

```
CREATE EXTERNAL TABLE `mypartitionedemrlogs`(
  `data` string COMMENT 'from deserializer')
 partitioned by (logtype string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n'
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6'
```

Les instructions de requête suivantes créent des partitions de table basées sur des sous-répertoires pour différents types de journaux créés par Amazon EMR dans Simple Storage Service (Amazon S3) :

```
ALTER TABLE mypartitionedemrlogs ADD
     PARTITION (logtype='containers')
     LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6/
containers/'
```

```
ALTER TABLE mypartitionedemrlogs ADD
     PARTITION (logtype='hadoop-mapreduce')
     LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6/
hadoop-mapreduce/'
```

```
ALTER TABLE mypartitionedemrlogs ADD
     PARTITION (logtype='hadoop-state-pusher')
     LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6/
hadoop-state-pusher/'
```

```
ALTER TABLE mypartitionedemrlogs ADD
```

```
     PARTITION (logtype='node')
     LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6/
node/'
```

```
ALTER TABLE mypartitionedemrlogs ADD
     PARTITION (logtype='steps')
     LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6/
steps/'
```

Après avoir créé les partitions, vous pouvez exécuter une requête `SHOW PARTITIONS` sur la table pour confirmer :

```
SHOW PARTITIONS mypartitionedemrlogs;
```

Les exemples suivants illustrent les requêtes pour des entrées de journal spécifiques utilisant la table et les partitions créées par les exemples ci-dessus.

Example – Interrogation des journaux d'application application_1561661818238_0002 dans la partition des conteneurs pour ERROR ou WARN

```
SELECT data,
        "$PATH"
FROM "default"."mypartitionedemrlogs"
WHERE logtype='containers'
        AND regexp_like("$PATH",'application_1561661818238_0002')
        AND regexp_like(data, 'ERROR|WARN') limit 100;
```

Example – Interrogation de la partition hadoop-Mapreduce pour la tâche job_1561661818238_0004 et des échecs de réduction

```
SELECT data,
        "$PATH"
FROM "default"."mypartitionedemrlogs"
WHERE logtype='hadoop-mapreduce'
        AND regexp_like(data,'job_1561661818238_0004|Failed Reduces') limit 100;
```

Example – Interrogation des journaux Hive dans la partition de nœud pour l'ID de requête 056e0609-33e1-4611-956c-7a31b42d2663

```
SELECT data,
```

```
        "$PATH"
FROM "default"."mypartitionedemrlogs"
WHERE logtype='node'
        AND regexp_like("$PATH",'hive')
        AND regexp_like(data,'056e0609-33e1-4611-956c-7a31b42d2663') limit 100;
```

Example – Interrogation des journaux resourcemanager dans la partition de nœud pour l'application 1567660019320_0001_01_000001

```
SELECT data,
        "$PATH"
FROM "default"."mypartitionedemrlogs"
WHERE logtype='node'
        AND regexp_like(data,'resourcemanager')
        AND regexp_like(data,'1567660019320_0001_01_000001') limit 100
```

Interrogation des journaux AWS Global Accelerator de flux

Vous pouvez les utiliser AWS Global Accelerator pour créer des accélérateurs qui dirigent le trafic réseau vers des points de terminaison optimaux sur le réseau AWS mondial. Pour plus d'informations sur Global Accelerator, voir Qu'est-ce que AWS Global Accelerator.

Les journaux de flux Global Accelerator vous permettent de collecter les informations relatives au trafic de l'adresse IP entrant et sortant dans les interfaces réseau de vos accélérateurs. Les données du journal de flux sont publiées dans Simple Storage Service (Amazon S3), où vous pouvez récupérer et consulter vos données. Pour plus d'informations, consultez Journaux de flux AWS Global Accelerator.

Vous pouvez utiliser Athena pour interroger vos journaux de flux Global Accelerator en créant une table qui spécifie leur emplacement dans Simple Storage Service (Amazon S3).

Création de la table pour les journaux de flux Global Accelerator

1. Copiez et collez l'instruction DDL suivante dans la console Athena. Cette requête spécifie ROW FORMAT DELIMITED et omet de spécifier a SerDe, ce qui signifie que la requête utilise le. `LazySimpleSerDe` Dans cette requête, les champs sont terminés par un espace.

   ```
   CREATE EXTERNAL TABLE IF NOT EXISTS aga_flow_logs (
     version string,
     account string,
     acceleratorid string,
   ```

```
  clientip string,
  clientport int,
  gip string,
  gipport int,
  endpointip string,
  endpointport int,
  protocol string,
  ipaddresstype string,
  numpackets bigint,
  numbytes int,
  starttime int,
  endtime int,
  action string,
  logstatus string,
  agasourceip string,
  agasourceport int,
  endpointregion string,
  agaregion string,
  direction string
)
PARTITIONED BY (dt string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ' '
LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/account_id/globalaccelerator/
region/'
TBLPROPERTIES ("skip.header.line.count"="1");
```

2. Modifiez la valeur `LOCATION` pour qu'elle pointe vers le compartiment Simple Storage Service (Amazon S3) qui contient vos données de journal.

```
's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/account_id/globalaccelerator/region_code/'
```

3. Exécutez la requête dans la console Athena. Une fois la requête terminée, Athena enregistre la table `aga_flow_logs` afin de rendre les données qu'elle contient disponibles pour les requêtes.
4. Créez des partitions pour lire les données, comme dans l'exemple de requête suivant. La requête suivant crée une partition unique pour une date spécifique. Remplacez les espaces réservés de date et d'emplacement.

```
ALTER TABLE aga_flow_logs
ADD PARTITION (dt='YYYY-MM-dd')
LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/account_id/
globalaccelerator/region_code/YYYY/MM/dd';
```

Exemples de requêtes pour les journaux AWS Global Accelerator de flux

Example – Répertorier les demandes qui passent par un emplacement périphérique spécifique

L'exemple de requête suivant répertorie les demandes passées par l'emplacement périphérique LHR. Utilisez l'opérateur `LIMIT` pour limiter le nombre de journaux à interroger à la fois.

```
SELECT
  clientip,
  agaregion,
  protocol,
  action
FROM
  aga_flow_logs
WHERE
  agaregion LIKE 'LHR%'
LIMIT
  100;
```

Example – Répertorier les adresses IP du point de terminaison qui reçoivent le plus grand nombre de requêtes HTTPS

Pour voir quelles adresses IP de point de terminaison reçoivent le plus grand nombre de requêtes HTTPS, utilisez la requête suivante. Elle comptabilise le nombre de paquets reçus sur le port HTTPS 443, les regroupe par adresse IP de destination et renvoie les 10 adresses IP principales.

```
SELECT
  SUM(numpackets) AS packetcount,
  endpointip
FROM
  aga_flow_logs
WHERE
  endpointport = 443
GROUP BY
  endpointip
ORDER BY
  packetcount DESC
LIMIT
  10;
```

Interroger les résultats d'Amazon GuardDuty

Amazon GuardDuty est un service de surveillance de la sécurité qui aide à identifier les activités inattendues, potentiellement non autorisées ou malveillantes dans votre AWS environnement. Lorsqu'il détecte une activité inattendue et potentiellement malveillante, il GuardDuty génère des résultats de sécurité que vous pouvez exporter vers Amazon S3 à des fins de stockage et d'analyse. Après avoir exporté vos résultats vers Simple Storage Service (Amazon S3), vous pouvez utiliser Athena pour les interroger. Cet article explique comment créer une table dans Athena pour vos GuardDuty résultats et les interroger.

Pour plus d'informations sur Amazon GuardDuty, consultez le guide de GuardDuty l'utilisateur Amazon.

Prérequis

- Activez la GuardDuty fonctionnalité d'exportation des résultats vers Amazon S3. Pour connaître les étapes à suivre, consultez la section Exportation des résultats dans le guide de GuardDuty l'utilisateur Amazon.

Création d'un tableau dans Athena pour les résultats GuardDuty

Pour interroger vos GuardDuty conclusions auprès d'Athéna, vous devez créer un tableau pour celles-ci.

Pour créer un tableau dans Athena pour les résultats GuardDuty

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Collez l'instruction DDL suivante dans la console Athena. Modifiez les valeurs `LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/GuardDuty/'` pour qu'elles correspondent à vos GuardDuty résultats dans Amazon S3.

```
CREATE EXTERNAL TABLE `gd_logs` (
  `schemaversion` string,
  `accountid` string,
  `region` string,
  `partition` string,
  `id` string,
  `arn` string,
  `type` string,
  `resource` string,
```

```
  `service` string,
  `severity` string,
  `createdat` string,
  `updatedat` string,
  `title` string,
  `description` string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
 LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/GuardDuty/'
 TBLPROPERTIES ('has_encrypted_data'='true')
```

> **Note**
>
> Il SerDe s'attend à ce que chaque document JSON se trouve sur une seule ligne de texte sans aucun caractère de fin de ligne séparant les champs de l'enregistrement. Si le texte JSON est dans un joli format d'impression, vous pouvez recevoir un message d'erreur tel que HIVE_CURSOR_ERROR : Row is not a valid JSON Object ou HIVE_CURSOR_ERROR : : Unexpected JsonParseException end-of-input : expected close marker for OBJECT lorsque vous essayez d'interroger la table après l'avoir créée. Pour plus d'informations, consultez la section Fichiers de données JSON dans la SerDe documentation OpenX sur. GitHub

3. Exécutez la requête dans la console Athena pour enregistrer la table `gd_logs`. Une fois la requête terminée, vous pouvez interroger les résultats à partir d'Athena.

Exemples de requêtes

Les exemples suivants montrent comment interroger les GuardDuty résultats d'Athéna.

Example – Exfiltration de données DNS

La requête suivante renvoie des informations sur les instances Amazon EC2 qui peuvent exfiltrer des données via des requêtes DNS.

```
SELECT
    title,
    severity,
    type,
    id AS FindingID,
    accountid,
    region,
```

```
    createdat,
    updatedat,
    json_extract_scalar(service, '$.count') AS Count,
    json_extract_scalar(resource, '$.instancedetails.instanceid') AS InstanceID,
    json_extract_scalar(service, '$.action.actiontype') AS DNS_ActionType,
    json_extract_scalar(service, '$.action.dnsrequestaction.domain') AS DomainName,
    json_extract_scalar(service, '$.action.dnsrequestaction.protocol') AS protocol,
    json_extract_scalar(service, '$.action.dnsrequestaction.blocked') AS blocked
FROM gd_logs
WHERE type = 'Trojan:EC2/DNSDataExfiltration'
ORDER BY severity DESC
```

Example – Accès utilisateur IAM non autorisé

La requête suivante renvoie tous les types de résultat `UnauthorizedAccess:IAMUser` pour un principal IAM à partir de toutes les régions.

```
SELECT title,
         severity,
         type,
         id,
         accountid,
         region,
         createdat,
         updatedat,
         json_extract_scalar(service, '$.count') AS Count,
         json_extract_scalar(resource, '$.accesskeydetails.username') AS IAMPrincipal,
         json_extract_scalar(service,'$.action.awsapicallaction.api') AS
 APIActionCalled
FROM gd_logs
WHERE type LIKE '%UnauthorizedAccess:IAMUser%'
ORDER BY severity desc;
```

Conseils pour interroger GuardDuty les résultats

Lorsque vous créez votre requête, gardez les points suivants à l'esprit.

- Pour extraire des données à partir de champs JSON imbriqués, utilisez les fonctions Presto `json_extract_scalar` ou `json_extract`. Pour plus d'informations, consultez Extraction de données JSON à partir de chaînes.
- Assurez-vous que tous les caractères des champs JSON sont en minuscules.

- Pour plus d'informations sur le téléchargement des résultats de requête, veuillez consulter Téléchargement des fichiers de résultats de requête à l'aide de la console Athena.

Journaux d'interrogation AWS Network Firewall

AWS Network Firewall est un service géré que vous pouvez utiliser pour déployer des protections réseau essentielles pour vos instances Amazon Virtual Private Cloud. AWS Network Firewall fonctionne de concert avec AWS Firewall Manager ce qui vous permet de créer des politiques basées sur des AWS Network Firewall règles, puis de les appliquer de manière centralisée à l'ensemble de vos VPC et de vos comptes. Pour plus d'informations sur AWS Network Firewall, voir AWS Network Firewall.

Vous pouvez configurer la AWS Network Firewall journalisation du trafic que vous transférez vers le moteur de règles dynamiques de votre pare-feu. La journalisation vous fournit des informations détaillées sur le trafic réseau, notamment l'heure à laquelle le moteur avec état a reçu un paquet, des informations détaillées sur le paquet et toute action de règle avec état prise à l'encontre du paquet. Les journaux sont publiés dans la destination des journaux que vous avez configurée, où vous pouvez les récupérer et les consulter. Pour plus d'informations, consultez la rubrique Journalisation du trafic réseau à partir du AWS Network Firewall du Guide du développeur AWS Network Firewall .

Création d'un tableau pour les journaux d'alertes

1. Modifiez l'exemple d'instruction DDL suivant pour le conformer à la structure de votre journal d'alertes. Il se peut que vous deviez mettre à jour l'instruction pour inclure les colonnes de la dernière version des journaux. Pour plus d'informations, consultez la rubrique Contenu d'un journal de pare-feu du Guide du développeur AWS Network Firewall .

```
CREATE EXTERNAL TABLE network_firewall_alert_logs (
  firewall_name string,
  availability_zone string,
  event_timestamp string,
  event struct<
    timestamp:string,
    flow_id:bigint,
    event_type:string,
    src_ip:string,
    src_port:int,
    dest_ip:string,
    dest_port:int,
    proto:string,
```

```
    app_proto:string,
    tls_inspected:boolean,
    alert:struct<
      alert_id:string,
      alert_type:string,
      action:string,
      signature_id:int,
      rev:int,
      signature:string,
      category:string,
      severity:int,
      rule_name:string,
      alert_name:string,
      alert_severity:string,
      alert_description:string,
      file_name:string,
      file_hash:string,
      packet_capture:string,
      reference_links:array<string>
    >,
    src_country:string,
    dest_country:string,
    src_hostname:string,
    dest_hostname:string,
    user_agent:string,
    url:string
   >
)
 ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
 LOCATION 's3://DOC-EXAMPLE-BUCKET/path_to_alert_logs_folder/';
```

2. Modifiez la `LOCATION` clause pour spécifier le dossier de vos journaux dans Amazon S3.
3. Exécutez votre `CREATE TABLE` requête dans l'éditeur de requêtes Athena. Une fois la requête terminée, Athena enregistre la `network_firewall_alert_logs` table et prépare les données vers lesquelles elle pointe pour les requêtes.

Exemple de requête dans le journal d'alertes

L'exemple de requête du journal d'alertes présenté dans cette section filtre les événements au cours desquels une inspection TLS a été effectuée et qui comportent des alertes d'un niveau de gravité supérieur ou égal à 2.

La requête utilise des alias pour créer des en-têtes de colonne de sortie indiquant à `struct` quoi appartient la colonne. Par exemple, l'en-tête de colonne du `event.alert.category` champ est `event_alert_category` au lieu de simplement`category`. Pour personnaliser davantage les noms des colonnes, vous pouvez modifier les alias en fonction de vos préférences. Par exemple, vous pouvez utiliser des traits de soulignement ou d'autres séparateurs pour délimiter les `struct` noms et les noms de champs.

N'oubliez pas de modifier les noms et les `struct` références des colonnes en fonction de la définition de votre table et des champs que vous souhaitez voir apparaître dans le résultat de la requête.

```
SELECT
  firewall_name,
  availability_zone,
  event_timestamp,
  event.timestamp AS event_timestamp,
  event.flow_id AS event_flow_id,
  event.event_type AS event_type,
  event.src_ip AS event_src_ip,
  event.src_port AS event_src_port,
  event.dest_ip AS event_dest_ip,
  event.dest_port AS event_dest_port,
  event.proto AS event_protol,
  event.app_proto AS event_app_proto,
  event.tls_inspected AS event_tls_inspected,
  event.alert.alert_id AS event_alert_alert_id,
  event.alert.alert_type AS event_alert_alert_type,
  event.alert.action AS event_alert_action,
  event.alert.signature_id AS event_alert_signature_id,
  event.alert.rev AS event_alert_rev,
  event.alert.signature AS event_alert_signature,
  event.alert.category AS event_alert_category,
  event.alert.severity AS event_alert_severity,
  event.alert.rule_name AS event_alert_rule_name,
  event.alert.alert_name AS event_alert_alert_name,
  event.alert.alert_severity AS event_alert_alert_severity,
  event.alert.alert_description AS event_alert_alert_description,
  event.alert.file_name AS event_alert_file_name,
  event.alert.file_hash AS event_alert_file_hash,
  event.alert.packet_capture AS event_alert_packet_capture,
  event.alert.reference_links AS event_alert_reference_links,
  event.src_country AS event_src_country,
```

```
  event.dest_country AS event_dest_country,
  event.src_hostname AS event_src_hostname,
  event.dest_hostname AS event_dest_hostname,
  event.user_agent AS event_user_agent,
  event.url AS event_url
FROM
  network_firewall_alert_logs
WHERE
  event.alert.severity >= 2
  AND event.tls_inspected = true
LIMIT 10;
```

Création d'une table pour les journaux Netflow

1. Modifiez l'exemple d'instruction DDL suivant pour vous conformer à la structure de vos journaux Netflow. Il se peut que vous deviez mettre à jour l'instruction pour inclure les colonnes de la dernière version des journaux. Pour plus d'informations, consultez la rubrique Contenu d'un journal de pare-feu du Guide du développeur AWS Network Firewall .

```
CREATE EXTERNAL TABLE network_firewall_netflow_logs (
  firewall_name string,
  availability_zone string,
  event_timestamp string,
  event struct<
    timestamp:string,
    flow_id:bigint,
    event_type:string,
    src_ip:string,
    src_port:int,
    dest_ip:string,
    dest_port:int,
    proto:string,
    app_proto:string,
    netflow:struct<
      pkts:int,
      bytes:bigint,
      start:string,
      `end`:string,
      age:int,
      min_ttl:int,
      max_ttl:int,
      tcp_flags:struct<
        syn:boolean,
```

```
        fin:boolean,
        rst:boolean,
        psh:boolean,
        ack:boolean,
        urg:boolean
      >,
      tls_inspected:boolean
    >
  >
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/path_to_netflow_logs_folder/';
```

2. Modifiez la `LOCATION` clause pour spécifier le dossier de vos journaux dans Amazon S3.
3. Exécutez la `CREATE TABLE` requête dans l'éditeur de requêtes Athena. Une fois la requête terminée, Athena enregistre la `network_firewall_netflow_logs` table et prépare les données vers lesquelles elle pointe pour les requêtes.

Exemple de requête dans le journal Netflow

L'exemple de requête du journal Netflow présenté dans cette section filtre les événements au cours desquels une inspection TLS a été effectuée.

La requête utilise des alias pour créer des en-têtes de colonne de sortie indiquant à `struct` quoi appartient la colonne. Par exemple, l'en-tête de colonne du `event.netflow.bytes` champ est `event_netflow_bytes` au lieu de simplement`bytes`. Pour personnaliser davantage les noms des colonnes, vous pouvez modifier les alias en fonction de vos préférences. Par exemple, vous pouvez utiliser des traits de soulignement ou d'autres séparateurs pour délimiter les `struct` noms et les noms de champs.

N'oubliez pas de modifier les noms et les `struct` références des colonnes en fonction de la définition de votre table et des champs que vous souhaitez voir apparaître dans le résultat de la requête.

```
SELECT
  event.src_ip AS event_src_ip,
  event.dest_ip AS event_dest_ip,
  event.proto AS event_proto,
  event.app_proto AS event_app_proto,
  event.netflow.pkts AS event_netflow_pkts,
```

```
  event.netflow.bytes AS event_netflow_bytes,
  event.netflow.tcp_flags.syn AS event_netflow_tcp_flags_syn,
  event.netflow.tls_inspected AS event_netflow_tls_inspected
FROM network_firewall_netflow_logs
WHERE event.netflow.tls_inspected = true
```

Interrogation des journaux du dispositif du Network Load Balancer

Utilisez Athena pour analyser et traiter les journaux du Network Load Balancer. Ces journaux reçoivent des informations détaillées sur les demandes TLS (Transport Layer Security, Sécurité de la couche de transport) envoyées au Network Load Balancer. Vous pouvez utiliser ces journaux d'accès pour analyser les modèles de trafic et résoudre des problèmes.

Avant d'analyser les journaux d'accès du Network Load Balancer, activez-les et configurez-les pour les enregistrer dans le compartiment Simple Storage Service (Amazon S3) de destination. Pour plus d'informations ainsi que pour des informations sur chaque entrée du journal d'accès de Network Load Balancer, consultez Journaux d'accès de votre Network Load Balancer.

- Création de la table pour les journaux du Network Load Balancer
- Exemples de requêtes du Network Load Balancer

Créer la table pour les journaux du Network Load Balancer

1. Copiez et collez l'instruction DDL suivante dans la console Athena. Vérifiez la syntaxe des registres du journal du Network Load Balancer. Il se peut que vous ayez besoin de mettre à jour la requête suivante afin d'inclure les colonnes et la syntaxe d'expression régulière (Regex) pour la dernière version de l'enregistrement.

```
CREATE EXTERNAL TABLE IF NOT EXISTS nlb_tls_logs (
            type string,
            version string,
            time string,
            elb string,
            listener_id string,
            client_ip string,
            client_port int,
            target_ip string,
            target_port int,
            tcp_connection_time_ms double,
            tls_handshake_time_ms double,
```

```
            received_bytes bigint,
            sent_bytes bigint,
            incoming_tls_alert int,
            cert_arn string,
            certificate_serial string,
            tls_cipher_suite string,
            tls_protocol_version string,
            tls_named_group string,
            domain_name string,
            alpn_fe_protocol string,
            alpn_be_protocol string,
            alpn_client_preference_list string,
            tls_connection_creation_time string
            )
            ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
            WITH SERDEPROPERTIES (
            'serialization.format' = '1',
            'input.regex' =
        '([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*):([0-9]*)
 ([-.0-9]*) ([-.0-9]*) ([-0-9]*) ([-0-9]*) ([-0-9]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)$')
            LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/AWS_account_ID/
elasticloadbalancing/region';
```

2. Modifiez le compartiment Simple Storage Service (Amazon S3) `LOCATION` pour spécifier la destination des journaux du Network Load Balancer.
3. Exécutez la requête dans la console Athena. Une fois la requête terminée, Athena enregistre la table `nlb_tls_logs`, afin d'en préparer les données pour les requêtes.

Exemples de requêtes du Network Load Balancer

Pour voir le nombre de fois où un certificat est utilisé, utilisez une requête similaire à l'exemple suivant :

```
SELECT count(*) AS
         ct,
         cert_arn
FROM "nlb_tls_logs"
GROUP BY  cert_arn;
```

La requête suivante montre combien d'utilisateurs utilisent une version de TLS antérieure à 1.3 :

```
SELECT tls_protocol_version,
         COUNT(tls_protocol_version) AS
         num_connections,
         client_ip
FROM "nlb_tls_logs"
WHERE tls_protocol_version < 'tlsv13'
GROUP BY tls_protocol_version, client_ip;
```

Utilisez la requête suivante pour identifier les connexions qui prennent beaucoup de temps pour établir des liaisons TLS :

```
SELECT *
FROM "nlb_tls_logs"
ORDER BY  tls_handshake_time_ms DESC
LIMIT 10;
```

Utilisez la requête suivante pour identifier et compter les versions du protocole TLS et les suites de chiffrement négociées au cours des 30 derniers jours.

```
SELECT tls_cipher_suite,
         tls_protocol_version,
         COUNT(*) AS ct
FROM "nlb_tls_logs"
WHERE from_iso8601_timestamp(time) > current_timestamp - interval '30' day
        AND NOT tls_protocol_version = '-'
GROUP BY tls_cipher_suite, tls_protocol_version
ORDER BY ct DESC;
```

Interrogation des journaux de requête d'Amazon Route 53 Resolver

Vous pouvez créer des tables Athena pour vos journaux de requête Amazon Route 53 Resolver et les interroger à partir d'Athena.

La journalisation des requêtes Route 53 Resolver sert à la journalisation des requêtes DNS effectuées par des ressources au sein d'un VPC, des ressources sur site qui utilisent des points de terminaison du résolveur entrant, des requêtes qui utilisent un point de terminaison du résolveur sortant pour la résolution DNS récursive et des requêtes qui utilisent des règles de pare-feu DNS Route 53 Resolver pour bloquer, autoriser ou contrôler une liste de domaines. Pour plus d'informations sur la journalisation des requêtes du résolveur, consultez la rubrique Journalisation des requêtes du résolveur du Guide du développeur Amazon Route 53. Pour plus d'informations sur

chacun des champs des journaux, consultez la rubrique Valeurs apparaissant dans les journaux de requête du résolveur du Guide du développeur Amazon Route 53.

Création de la table pour les journaux de requête du résolveur

Vous pouvez utiliser l'éditeur de requête de la console Athena pour créer et interroger une table pour vos journaux de requêtes Route 53 Resolver.

Création et interrogation d'une table Athena pour les journaux de requêtes Route 53 Resolver

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans l'éditeur de requêtes Athena, saisissez l'instruction `CREATE TABLE` suivante. Remplacez les valeurs de la clause `LOCATION` par celles correspondant à l'emplacement de vos journaux Resolver dans Simple Storage Service (Amazon S3).

```
CREATE EXTERNAL TABLE r53_rlogs (
  version string,
  account_id string,
  region string,
  vpc_id string,
  query_timestamp string,
  query_name string,
  query_type string,
  query_class
    string,
  rcode string,
  answers array<
    struct<
      Rdata: string,
      Type: string,
      Class: string>
    >,
  srcaddr string,
  srcport int,
  transport string,
  srcids struct<
    instance: string,
    resolver_endpoint: string
    >,
  firewall_rule_action string,
  firewall_rule_group_id string,
  firewall_domain_list_id string
 )
```

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/aws_account_id/vpcdnsquerylogs/{vpc-id}/'
```

Les données du journal des requêtes Resolver étant au format JSON, l'instruction CREATE TABLE utilise une SerDebibliothèque JSON pour analyser les données.

> **Note**
>
> Il SerDe s'attend à ce que chaque document JSON se trouve sur une seule ligne de texte sans aucun caractère de fin de ligne séparant les champs de l'enregistrement. Si le texte JSON est dans un joli format d'impression, vous pouvez recevoir un message d'erreur tel que HIVE_CURSOR_ERROR : Row is not a valid JSON Object ou HIVE_CURSOR_ERROR : : Unexpected JsonParseException end-of-input : expected close marker for OBJECT lorsque vous essayez d'interroger la table après l'avoir créée. Pour plus d'informations, consultez la section Fichiers de données JSON dans la SerDe documentation OpenX sur. GitHub

3. Choisissez Exécuter la requête. L'instruction crée une table Athena nommée `r53_rlogs` dont les colonnes représentent chacun des champs de vos données de journal du résolveur.
4. Dans l'éditeur de requête de la console Athena, exécutez la requête suivante pour vérifier que votre table a été créée.

```
SELECT * FROM "r53_rlogs" LIMIT 10
```

Exemple de partitionnement

L'exemple suivant montre une instruction `CREATE TABLE` pour les journaux de requête Resolver qui utilise la projection de partition et est partitionnée par VPC et par date. Pour plus d'informations sur la projection de partition, voir Projection de partition avec Amazon Athena.

```
CREATE EXTERNAL TABLE r53_rlogs (
  version string,
  account_id string,
  region string,
  vpc_id string,
  query_timestamp string,
  query_name string,
```

```
  query_type string,
  query_class string,
  rcode string,
  answers array<
    struct<
      Rdata: string,
      Type: string,
      Class: string>
    >,
  srcaddr string,
  srcport int,
  transport string,
  srcids struct<
    instance: string,
    resolver_endpoint: string
    >,
  firewall_rule_action string,
  firewall_rule_group_id string,
  firewall_domain_list_id string
)
PARTITIONED BY (
`date` string,
`vpc` string
)
ROW FORMAT SERDE      'org.openx.data.jsonserde.JsonSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT          'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION              's3://DOC-EXAMPLE-BUCKET/route53-query-logging/
AWSLogs/aws_account_id/vpcdnsquerylogs/'
TBLPROPERTIES(
'projection.enabled' = 'true',
'projection.vpc.type' = 'enum',
'projection.vpc.values' = 'vpc-6446ae02',
'projection.date.type' = 'date',
'projection.date.range' = '2023/06/26,NOW',
'projection.date.format' = 'yyyy/MM/dd',
'projection.date.interval' = '1',
'projection.date.interval.unit' = 'DAYS',
'storage.location.template' = 's3://DOC-EXAMPLE-BUCKET/route53-query-logging/
AWSLogs/aws_account_id/vpcdnsquerylogs/${vpc}/${date}/'
)
```

Exemples de requêtes

Les exemples suivants illustrent certaines requêtes que vous pouvez effectuer à partir d'Athena sur vos journaux de requêtes Resolver.

Exemple 1 - Interrogation des journaux dans l'ordre query_timestamp décroissant

La requête suivante affiche les résultats du journal par ordre `query_timestamp` décroissant.

```
SELECT * FROM "r53_rlogs"
ORDER BY query_timestamp DESC
```

Exemple 2 - Interrogation des journaux entre les heures de début et de fin spécifiées

La requête suivante interroge les journaux entre minuit et 8 heures du matin le 24 septembre 2020. Remplacez les heures de début et de fin selon vos besoins.

```
SELECT query_timestamp, srcids.instance, srcaddr, srcport, query_name, rcode
FROM "r53_rlogs"
WHERE (parse_datetime(query_timestamp,'yyyy-MM-dd''T''HH:mm:ss''Z')
     BETWEEN parse_datetime('2020-09-24-00:00:00','yyyy-MM-dd-HH:mm:ss')
     AND parse_datetime('2020-09-24-00:08:00','yyyy-MM-dd-HH:mm:ss'))
ORDER BY query_timestamp DESC
```

Exemple 3 - Interrogation des journaux en fonction d'un modèle de nom de requête DNS spécifié

La requête suivante sélectionne les registres dont le nom de requête inclut la chaîne « example.com ».

```
SELECT query_timestamp, srcids.instance, srcaddr, srcport, query_name, rcode, answers
FROM "r53_rlogs"
WHERE query_name LIKE '%example.com%'
ORDER BY query_timestamp DESC
```

Exemple 4 - Requêtes d'interrogation des journaux sans réponse

La requête suivante sélectionne les entrées du journal dans lesquelles la demande n'a pas reçu de réponse.

```
SELECT query_timestamp, srcids.instance, srcaddr, srcport, query_name, rcode, answers
FROM "r53_rlogs"
```

```
WHERE cardinality(answers) = 0
```

Exemple 5 - Interrogation des journaux avec une réponse spécifique

La requête suivante montre les journaux dans lesquels la valeur `answer.Rdata` a l'adresse IP spécifiée.

```
SELECT query_timestamp, srcids.instance, srcaddr, srcport, query_name, rcode,
 answer.Rdata
FROM "r53_rlogs"
CROSS JOIN UNNEST(r53_rlogs.answers) as st(answer)
WHERE answer.Rdata='203.0.113.16';
```

Interrogation des journaux d'événements Amazon SES

Vous pouvez utiliser Amazon Athena pour interroger les journaux d'événements Amazon Simple Email Service (Amazon SES).

Amazon SES est une plateforme de messagerie qui offre un moyen pratique et économique d'envoyer et de recevoir du courrier électronique en utilisant vos propres adresses e-mail et domaines. Vous pouvez surveiller votre activité d'envoi Amazon SES à un niveau précis à l'aide d'événements, de métriques et de statistiques.

Sur la base des caractéristiques que vous définissez, vous pouvez publier des événements Amazon SES sur Amazon CloudWatch, Amazon Data Firehose ou Amazon Simple Notification Service. Après le stockage des informations dans Amazon S3, vous pouvez les interroger à partir d'Amazon Athena.

Pour un exemple de `CREATE TABLE` déclaration Athena pour les journaux d'événements Amazon SES, y compris les étapes à suivre pour créer des vues et aplatir des tableaux imbriqués dans les données des journaux d'événements Amazon SES, consultez « Étape 3 : Utilisation d'Amazon Athena pour interroger les journaux d'événements SES » dans le AWS billet de blog Analyzing Amazon SES event data with Analytics Services. AWS

Interrogation des journaux de flux Amazon VPC

Les journaux de flux de cloud privé virtuel Amazon Virtual Private Cloud capturent des informations sur le trafic IP circulant vers et depuis les interfaces réseau d'un VPC. Utilisez les journaux pour examiner les modèles de trafic réseau, et identifier les menaces et les risques au sein de votre réseau VPC.

Pour interroger vos journaux de flux Amazon VPC, deux options s'offrent à vous :

- Console Amazon VPC : utilisez la fonctionnalité d'intégration Athena de la console Amazon VPC pour générer un modèle AWS CloudFormation qui crée une base de données Athena, un groupe de travail et une table de journaux de flux avec partitionnement pour vous. Le modèle crée également un ensemble de requêtes de journaux de flux prédéfinies que vous pouvez utiliser pour obtenir des informations concernant le trafic qui passe par votre VPC.

  Pour de plus amples d'informations, consultez la section Interroger des journaux de flux à l'aide d'Amazon Athena dans le Guide de l'utilisateur Amazon VPC.
- Console Amazon Athena – Créez vos tables et vos requêtes directement dans la console Athena. Pour plus d'informations, continuez à lire cette page.

Création et interrogation de tables pour les journaux de flux VPC personnalisés

Avant de commencer à interroger les journaux dans Athena, activez les journaux de flux VPC et configurez-les pour qu'ils soient enregistrés dans votre compartiment Simple Storage Service (Amazon S3). Une fois que vous avez créé les journaux, laissez-les s'exécuter quelques minutes pour collecter les données. Les journaux sont créés dans un format de compression GZIP qu'Athena vous permet d'interroger directement.

Lorsque vous créez un journal de flux VPC personnalisé, vous pouvez utiliser un format personnalisé quand vous voulez spécifier les champs à renvoyer dans le journal de flux et l'ordre dans lequel ils doivent apparaître. Pour plus d'informations sur les enregistrements des journaux de flux, consultez Enregistrements des journaux de flux du Guide de l'utilisateur Amazon VPC.

Considérations communes

Lorsque vous créez des tables dans les journaux de flux Athena pour Amazon VPC, n'oubliez pas les points suivants :

- Par défaut, sur Athena, Parquet accède aux colonnes par nom. Pour plus d'informations, consultez Traitement des mises à jour de schéma.
- Utilisez les noms des enregistrements du journal de flux pour les noms de colonnes sur Athena. Les noms des colonnes du schéma Athena doivent correspondre exactement aux noms des champs dans les journaux de flux Amazon VPC, avec les différences suivantes :
  - Remplacez les traits d'union dans les noms des champs de journaux Amazon VPC par des traits de soulignement dans les noms des colonnes Athena. Sur Athena, les seuls caractères acceptables pour les noms de bases de données, les noms de tables et les noms de colonnes

sont les lettres minuscules, les chiffres et le caractère de soulignement. Pour plus d'informations, consultez Noms de bases de données, de tables et de colonnes.

- Échappez les noms des enregistrements du journal de flux qui sont Mots-clés réservés sur Athena en les entourant de barres obliques inversées.

- Les journaux de flux VPC sont Compte AWS spécifiques. Lorsque vous publiez vos fichiers journaux sur Amazon S3, le chemin créé par Amazon VPC dans Amazon S3 inclut l'ID du Compte AWS utilisé pour créer le journal de flux. Pour plus d'informations, consultez Publier les journaux de flux vers Amazon S3 dans le Guide de l'utilisateur Amazon VPC.

Instruction CREATE TABLE pour les journaux de flux Amazon VPC

La procédure suivante permet de créer une table Amazon VPC pour les journaux de flux VPC. Lorsque vous créez un journal de flux avec un format personnalisé, vous créez une table avec des champs correspondant à ceux que vous avez spécifiés lors de la création du journal de flux, dans le même ordre que celui où vous les avez spécifiés.

Pour créer une table Athena pour les journaux de flux Amazon VPC

1. Saisissez une instruction DDL telle que la suivante dans l'éditeur de requêtes de la console Athena, en suivant les instructions de la section Considérations communes. L'exemple d'instruction suivant crée une table comportant les colonnes des journaux de flux Amazon VPC versions 2 à 5, comme indiqué dans la rubrique Registres des journaux de flux. Si vous utilisez un autre jeu de colonnes ou un autre ordre de colonnes, modifiez l'instruction en conséquence.

```
CREATE EXTERNAL TABLE IF NOT EXISTS `vpc_flow_logs` (
  version int,
  account_id string,
  interface_id string,
  srcaddr string,
  dstaddr string,
  srcport int,
  dstport int,
  protocol bigint,
  packets bigint,
  bytes bigint,
  start bigint,
  `end` bigint,
  action string,
  log_status string,
  vpc_id string,
```

```
  subnet_id string,
  instance_id string,
  tcp_flags int,
  type string,
  pkt_srcaddr string,
  pkt_dstaddr string,
  region string,
  az_id string,
  sublocation_type string,
  sublocation_id string,
  pkt_src_aws_service string,
  pkt_dst_aws_service string,
  flow_direction string,
  traffic_path int
)
PARTITIONED BY (`date` date)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ' '
LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/{account_id}/
vpcflowlogs/{region_code}/'
TBLPROPERTIES ("skip.header.line.count"="1");
```

Notez les points suivants :

- La requête spécifie `ROW FORMAT DELIMITED` et omet de spécifier un SerDe. Cela signifie que la requête utilise le LazySimpleSerDe pour les fichiers CSV, TSV et délimités de manière personnalisée. Dans cette requête, les champs sont terminés par un espace.
- La clause `PARTITIONED BY` utilise le type `date`. Cela permet d'utiliser des opérateurs mathématiques dans les requêtes pour sélectionner ce qui est plus ancien ou plus récent par rapport à une certaine date.

  > **Note**
  >
  > Étant donné que `date` est un mot-clé réservé dans les instructions DDL, il est entouré de guillemets inversés. Pour plus d'informations, consultez Mots-clés réservés.

- Pour un journal de flux VPC avec un autre format personnalisé, modifiez les champs pour qu'ils correspondent à ceux que vous avez spécifiés lors de la création du journal de flux.

2. Modifiez le `LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/{account_id}/vpcflowlogs/{region_code}/'` afin de pointer vers le compartiment Amazon S3 qui contient les données de vos journaux.
3. Exécutez la requête dans la console Athena. Une fois que la requête est terminée, Athena enregistre la table `vpc_flow_logs`, de telle sorte que les données soient prêtes pour que vous puissiez émettre des requêtes.
4. Créez des partitions pour être en mesure de lire les données, comme dans l'exemple de requête suivant. Cette requête suivant crée une partition unique pour une date spécifique. Remplacez les espaces réservés de date et d'emplacement en fonction des besoins.

> **Note**
>
> Cette requête crée seulement une partition unique, pour une date que vous spécifiez. Pour automatiser le processus, utilisez un script qui exécute cette requête et crée des partitions de cette manière pour la `year/month/day`, ou utilisez une instruction `CREATE TABLE` qui spécifie la projection des partitions.

```
ALTER TABLE vpc_flow_logs
ADD PARTITION (`date`='YYYY-MM-dd')
LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/{account_id}/
vpcflowlogs/{region_code}/YYYY/MM/dd';
```

Exemples de requêtes pour la table vpc_flow_logs

Utilisez l'éditeur de requêtes de la console Athena pour exécuter des instructions SQL sur la table que vous créez. Vous pouvez enregistrer les requêtes, afficher des requêtes précédentes ou télécharger les résultats de la requête au format CSV. Dans les exemples suivants, remplacer `vpc_flow_logs` par le nom de votre table. Modifiez les valeurs de colonne et d'autres variables en fonction de vos besoins.

L'exemple de requête suivant répertorie un maximum de 100 journaux de flux pour la date spécifiée.

```
SELECT *
FROM vpc_flow_logs
WHERE date = DATE('2020-05-04')
LIMIT 100;
```

La requête suivante répertorie toutes les connexions TCP rejetées et utilise la nouvelle colonne de partition de date, `date`, pour en extraire le jour de la semaine pendant lequel ces événements sont survenus.

```
SELECT day_of_week(date) AS
  day,
  date,
  interface_id,
  srcaddr,
  action,
  protocol
FROM vpc_flow_logs
WHERE action = 'REJECT' AND protocol = 6
LIMIT 100;
```

Pour voir lequel de vos serveurs reçoit le plus grand nombre de demandes HTTPS, utilisez cette requête. Elle comptabilise le nombre de paquets reçus sur le port HTTPS 443, les regroupe par adresse IP de destination et renvoie les 10 principaux.

```
SELECT SUM(packets) AS
  packetcount,
  dstaddr
FROM vpc_flow_logs
WHERE dstport = 443 AND date > current_date - interval '7' day
GROUP BY dstaddr
ORDER BY packetcount DESC
LIMIT 10;
```

Création de tables pour les journaux de flux au format Apache Parquet

La procédure suivante permet de créer une table Amazon VPC pour les journaux de flux VPC au format Apache Parquet.

Pour créer une table Athena pour les journaux de flux Amazon VPC au format Parquet

1. Saisissez une instruction DDL telle que la suivante dans l'éditeur de requêtes de la console Athena, en suivant les instructions de la section Considérations communes. L'exemple d'instruction suivant crée une table comportant les colonnes des journaux de flux Amazon VPC versions 2 à 5, comme indiqué dans la rubrique Registres des journaux de flux au format Parquet, Hive partitionné heure par heure. Si vous n'avez aucune partition horaire, retirez `hour` de la clause `PARTITIONED BY`.

```
CREATE EXTERNAL TABLE IF NOT EXISTS vpc_flow_logs_parquet (
  version int,
  account_id string,
  interface_id string,
  srcaddr string,
  dstaddr string,
  srcport int,
  dstport int,
  protocol bigint,
  packets bigint,
  bytes bigint,
  start bigint,
  `end` bigint,
  action string,
  log_status string,
  vpc_id string,
  subnet_id string,
  instance_id string,
  tcp_flags int,
  type string,
  pkt_srcaddr string,
  pkt_dstaddr string,
  region string,
  az_id string,
  sublocation_type string,
  sublocation_id string,
  pkt_src_aws_service string,
  pkt_dst_aws_service string,
  flow_direction string,
  traffic_path int
)
PARTITIONED BY (
  `aws-account-id` string,
  `aws-service` string,
  `aws-region` string,
  `year` string,
  `month` string,
  `day` string,
  `hour` string
)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
```

```
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/'
TBLPROPERTIES (
  'EXTERNAL'='true',
  'skip.header.line.count'='1'
  )
```

2. Modifiez l'exemple `LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/'` afin de pointer vers le chemin Simple Storage Service (Amazon S3) qui contient les données de vos journaux.
3. Exécutez la requête dans la console Athena.
4. Si vos données sont au format compatible Hive, exécutez la commande suivante dans la console Athena pour mettre à jour et charger les partitions Hive dans le métastore. Une fois la requête terminée, vous pouvez interroger les données dans la table `vpc_flow_logs_parquet`.

   ```
   MSCK REPAIR TABLE vpc_flow_logs_parquet
   ```

   Si vous n'utilisez pas de données compatibles avec Hive, exécutez ALTER TABLE ADD PARTITION pour charger les partitions.

Pour de plus amples informations sur l'utilisation d'Athena pour interroger les journaux de flux Amazon VPC au format Parquet, veuillez consulter l'article Optimiser les performances et réduire les coûts d'analytique réseau avec les journaux de flux VPC au format Apache Parquet dans le Blog Big Data AWS .

Création et interrogation d'une table pour les journaux de flux Amazon VPC à l'aide de la projection de partitions

Utilisez une instruction `CREATE TABLE` comme la suivante pour créer une table, partitionner la table et remplir automatiquement les partitions en utilisant projection de partition. Remplacer le nom de la table `test_table_vpclogs` dans l'exemple par le nom de votre table. Modifiez la clause `LOCATION` pour spécifier le compartiment Amazon S3 qui contient les données de vos journaux Amazon VPC.

L'instruction `CREATE TABLE` suivante concerne les journaux de flux VPC livrés dans un format de partitionnement de style non Hive. L'exemple permet l'agrégation de plusieurs comptes. Si vous

centralisez les journaux de flux VPC de plusieurs comptes dans un compartiment Amazon S3, l'ID du compte doit être saisi dans le chemin d'accès Amazon S3.

```
CREATE EXTERNAL TABLE IF NOT EXISTS test_table_vpclogs (
  version int,
  account_id string,
  interface_id string,
  srcaddr string,
  dstaddr string,
  srcport int,
  dstport int,
  protocol bigint,
  packets bigint,
  bytes bigint,
  start bigint,
  `end` bigint,
  action string,
  log_status string,
  vpc_id string,
  subnet_id string,
  instance_id string,
  tcp_flags int,
  type string,
  pkt_srcaddr string,
  pkt_dstaddr string,
  az_id string,
  sublocation_type string,
  sublocation_id string,
  pkt_src_aws_service string,
  pkt_dst_aws_service string,
  flow_direction string,
  traffic_path int
)
PARTITIONED BY (accid string, region string, day string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ' '
LOCATION '$LOCATION_OF_LOGS'
TBLPROPERTIES
(
"skip.header.line.count"="1",
"projection.enabled" = "true",
"projection.accid.type" = "enum",
"projection.accid.values" = "$ACCID_1,$ACCID_2",
```

```
"projection.region.type" = "enum",
"projection.region.values" = "$REGION_1,$REGION_2,$REGION_3",
"projection.day.type" = "date",
"projection.day.range" = "$START_RANGE,NOW",
"projection.day.format" = "yyyy/MM/dd",
"storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/AWSLogs/${accid}/vpcflowlogs/
${region}/${day}"
)
```

Exemples de requêtes pour la table test_table_vpclogs

L'exemple de requête suivant interroge la table `test_table_vpclogs` créée par l'instruction précédente `CREATE TABLE`. Remplacez `test_table_vpclogs` dans les requêtes par le nom de votre table. Modifiez les valeurs de colonne et d'autres variables en fonction de vos besoins.

Pour renvoyer les 100 premières entrées du journal d'accès dans l'ordre chronologique pendant une période donnée, exécutez une requête comme la suivante.

```
SELECT *
FROM test_table_vpclogs
WHERE day >= '2021/02/01' AND day < '2021/02/28'
ORDER BY day ASC
LIMIT 100
```

Pour voir quel serveur reçoit les dix premiers paquets HTTP pendant une période de temps spécifiée, exécutez une requête semblable à la suivante. La requête compte le nombre de paquets reçus sur le port HTTPS 443, les regroupe par adresse IP de destination et renvoie les 10 premières entrées de la semaine précédente.

```
SELECT SUM(packets) AS packetcount,
       dstaddr
FROM test_table_vpclogs
WHERE dstport = 443
  AND day >= '2021/03/01'
  AND day < '2021/03/31'
GROUP BY dstaddr
ORDER BY packetcount DESC
LIMIT 10
```

Pour renvoyer les journaux créés pendant une période donnée, exécutez une requête semblable à la suivante.

```
SELECT interface_id,
       srcaddr,
       action,
       protocol,
       to_iso8601(from_unixtime(start)) AS start_time,
       to_iso8601(from_unixtime("end")) AS end_time
FROM test_table_vpclogs
WHERE DAY >= '2021/04/01'
  AND DAY < '2021/04/30'
```

Pour renvoyer les journaux d'accès d'une adresse IP source entre des périodes spécifiées, exécutez une requête semblable à la suivante.

```
SELECT *
FROM test_table_vpclogs
WHERE srcaddr = '10.117.1.22'
  AND day >= '2021/02/01'
  AND day < '2021/02/28'
```

La requête ci-dessous affiche toutes les connexions TCP rejetées.

```
SELECT day,
       interface_id,
       srcaddr,
       action,
       protocol
FROM test_table_vpclogs
WHERE action = 'REJECT' AND protocol = 6 AND day >= '2021/02/01' AND day < '2021/02/28'
LIMIT 10
```

Pour renvoyer les journaux d'accès de la plage d'adresses IP commençant par `10.117`, exécutez une requête similaire à la suivante.

```
SELECT *
FROM test_table_vpclogs
WHERE split_part(srcaddr,'.', 1)='10'
  AND split_part(srcaddr,'.', 2) ='117'
```

Pour renvoyer les journaux d'accès d'une adresse IP de destination entre une certaine plage de temps, exécutez une requête semblable à la suivante.

```
SELECT *
FROM test_table_vpclogs
WHERE dstaddr = '10.0.1.14'
  AND day >= '2021/01/01'
  AND day < '2021/01/31'
```

Création de tables pour les journaux de flux au format Apache Parquet à l'aide de la projection de partition

L'instruction CREATE TABLE de projection de partition suivante pour les journaux de flux VPC est au format Apache Parquet, n'est pas compatible avec Hive et est partitionnée par heure et par date plutôt que par jour. Remplacer le nom de la table `test_table_vpclogs_parquet` dans l'exemple par le nom de votre table. Modifiez la clause `LOCATION` pour spécifier le compartiment Amazon S3 qui contient les données de vos journaux Amazon VPC.

```
CREATE EXTERNAL TABLE IF NOT EXISTS test_table_vpclogs_parquet (
  version int,
  account_id string,
  interface_id string,
  srcaddr string,
  dstaddr string,
  srcport int,
  dstport int,
  protocol bigint,
  packets bigint,
  bytes bigint,
  start bigint,
  `end` bigint,
  action string,
  log_status string,
  vpc_id string,
  subnet_id string,
  instance_id string,
  tcp_flags int,
  type string,
  pkt_srcaddr string,
  pkt_dstaddr string,
  az_id string,
  sublocation_type string,
  sublocation_id string,
  pkt_src_aws_service string,
  pkt_dst_aws_service string,
```

```
  flow_direction string,
  traffic_path int
)
PARTITIONED BY (region string, date string, hour string)
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT
'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/{account_id}/vpcflowlogs/'
TBLPROPERTIES (
"EXTERNAL"="true",
"skip.header.line.count" = "1",
"projection.enabled" = "true",
"projection.region.type" = "enum",
"projection.region.values" = "us-east-1,us-west-2,ap-south-1,eu-west-1",
"projection.date.type" = "date",
"projection.date.range" = "2021/01/01,NOW",
"projection.date.format" = "yyyy/MM/dd",
"projection.hour.type" = "integer",
"projection.hour.range" = "00,23",
"projection.hour.digits" = "2",
"storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/${account_id}/
vpcflowlogs/${region}/${date}/${hour}"
)
```

Ressources supplémentaires

Pour plus d'informations sur l'utilisation d'Athena pour analyser les journaux de flux VPC, consultez les articles suivants du blog AWS Big Data :

- Analysez les journaux de flux VPC grâce à l'intégration d'Amazon point-and-click Athena
- Analyse des journaux de flux VPC à l'aide d'Amazon Athena et Amazon QuickSight
- Optimiser les performances et réduire les coûts de l'analyse des réseaux grâce aux journaux de flux VPC au format Apache Parquet

Journaux d'interrogation AWS WAF

AWS WAF est un pare-feu d'applications Web qui vous permet de surveiller et de contrôler les requêtes HTTP et HTTPS que vos applications Web protégées reçoivent de la part des clients. Vous

définissez comment traiter les requêtes Web en configurant des règles dans une liste de contrôle d'accès AWS WAF Web (ACL). Vous protégez ensuite une application Web en lui associant une ACL Web. Parmi les ressources d'applications Web que vous pouvez protéger, AWS WAF citons les CloudFront distributions Amazon, les API REST Amazon API Gateway et les équilibreurs de charge d'application. Pour plus d'informations AWS WAF, consultez AWS WAFle guide du AWS WAF développeur.

AWS WAF les journaux incluent des informations sur le trafic analysé par votre ACL Web, telles que l'heure à laquelle la demande AWS WAF a été reçue de votre AWS ressource, des informations détaillées sur la demande et l'action pour la règle à laquelle chaque demande correspond.

Vous pouvez configurer une ACL AWS WAF Web pour publier les journaux vers l'une des nombreuses destinations, où vous pouvez les interroger et les consulter. Pour plus d'informations sur la configuration de la journalisation des ACL Web et du contenu des AWS WAF journaux, consultez la section Journalisation du trafic ACL AWS WAF Web dans le guide du AWS WAF développeur.

Pour savoir comment agréger les AWS WAF journaux dans un référentiel central de lacs de données et les interroger avec Athena, consultez le billet de blog sur le AWS Big Data Analyzing AWS WAF logs with OpenSearch Service, Amazon Athena et Amazon. QuickSight

Cette rubrique fournit deux exemples d'instructions `CREATE TABLE` : une qui utilise le partitionnement et une autre qui n'en utilise pas.

> **Note**
>
> Les instructions `CREATE TABLE` de cette rubrique peuvent être utilisées à la fois pour les journaux AWS WAF v1 et v2. Dans v1, le champ `webaclid` contient une ID. Dans v2, le champ `webaclid` contient un ARN complet. Les instructions `CREATE TABLE` ici traitent ce contenu de manière agnostique à l'aide du type de données `string`.

Rubriques

- Création d'une table pour les journaux AWS WAF S3 dans Athena à l'aide de la projection de partition
- Création d'une table pour les AWS WAF journaux sans partitionnement
- Exemples de requêtes pour les AWS WAF journaux

Création d'une table pour les journaux AWS WAF S3 dans Athena à l'aide de la projection de partition

Comme AWS WAF les journaux ont une structure connue dont vous pouvez spécifier le schéma de partition à l'avance, vous pouvez réduire le temps d'exécution des requêtes et automatiser la gestion des partitions en utilisant la fonction de projection de partition Athena. La projection des partitions ajoute automatiquement de nouvelles partitions à mesure que de nouvelles données sont ajoutées. Vous n'avez donc plus besoin d'ajouter manuellement des partitions à l'aide de la commande `ALTER TABLE ADD PARTITION`.

L'exemple d'`CREATE TABLE`instruction suivant utilise automatiquement la projection de partitions sur AWS WAF les journaux à partir d'une date spécifiée jusqu'à aujourd'hui pour quatre AWS régions différentes. La clause `PARTITION BY` dans cet exemple, partitionne par région et par date, mais vous pouvez modifier les partitions en fonction de vos besoins. Modifiez les champs si nécessaire pour qu'ils correspondent à la sortie de votre journal. Dans les `storage.location.template` clauses `LOCATION` et, remplacez les espaces réservés au *`compartiment`* et au *`AccountID`* par des valeurs qui identifient l'emplacement du compartiment Amazon S3 dans lequel se trouvent vos journaux. AWS WAF Pour `projection.day.range`, remplacez *`2021`*/*`01`*/*`01`* par la date de début que vous souhaitez utiliser. Après avoir exécuté la requête avec succès, vous pouvez interroger la table. Vous n'avez pas besoin d'exécuter `ALTER TABLE ADD PARTITION` pour charger les partitions.

```
CREATE EXTERNAL TABLE `waf_logs`(
  `timestamp` bigint,
  `formatversion` int,
  `webaclid` string,
  `terminatingruleid` string,
  `terminatingruletype` string,
  `action` string,
  `terminatingrulematchdetails` array <
                                    struct <
                                        conditiontype: string,
                                        sensitivitylevel: string,
                                        location: string,
                                        matcheddata: array < string >
                                          >
                                     >,
  `httpsourcename` string,
  `httpsourceid` string,
  `rulegrouplist` array <
                      struct <
                          rulegroupid: string,
```

```
                        terminatingrule: struct <
                                        ruleid: string,
                                        action: string,
                                        rulematchdetails: array <
                                                        struct <
                                                            conditiontype:
string,

sensitivitylevel: string,
                                                            location:
string,
                                                            matcheddata:
array < string >
                                                               >
                                                         >
                                      >,
                        nonterminatingmatchingrules: array <
                                                       struct <
                                                           ruleid: string,
                                                           action: string,
                                                           overriddenaction:
string,
                                                           rulematchdetails:
array <

struct <

   conditiontype: string,

   sensitivitylevel: string,

   location: string,

   matcheddata: array < string >

      >
                                                          >,
                                                           challengeresponse:
struct <

responsecode: string,

solvetimestamp: string
                                                                        >,
```

```
                                                                captcharesponse:
 struct <

 responsecode: string,

 solvetimestamp: string
                                                                              >
                                                                    >
                                                               >,
                          excludedrules: string
                            >
                       >,
`ratebasedrulelist` array <
                         struct <
                             ratebasedruleid: string,
                             limitkey: string,
                             maxrateallowed: int
                               >
                          >,
  `nonterminatingmatchingrules` array <
                                    struct <
                                        ruleid: string,
                                        action: string,
                                        rulematchdetails: array <
                                                              struct <
                                                                  conditiontype: string,
                                                                  sensitivitylevel:
 string,
                                                                  location: string,
                                                                  matcheddata: array <
 string >
                                                                     >
                                                              >,
                                        challengeresponse: struct <
                                                             responsecode: string,
                                                             solvetimestamp: string
                                                              >,
                                        captcharesponse: struct <
                                                             responsecode: string,
                                                             solvetimestamp: string
                                                              >
                                          >
                                     >,
  `requestheadersinserted` array <
```

```
                                struct <
                                    name: string,
                                    value: string
                                      >
                                 >,
  `responsecodesent` string,
  `httprequest` struct <
                    clientip: string,
                    country: string,
                    headers: array <
                                struct <
                                    name: string,
                                    value: string
                                      >
                                 >,
                    uri: string,
                    args: string,
                    httpversion: string,
                    httpmethod: string,
                    requestid: string
                      >,
  `labels` array <
              struct <
                  name: string
                    >
               >,
  `captcharesponse` struct <
                      responsecode: string,
                      solvetimestamp: string,
                      failureReason: string
                        >,
  `challengeresponse` struct <
                      responsecode: string,
                      solvetimestamp: string,
                      failureReason: string
                      >,
  `ja3Fingerprint` string,
  `oversizefields` string,
  `requestbodysize` int,
  `requestbodysizeinspectedbywaf` int
)
PARTITIONED BY (
`region` string,
`date` string)
```

```
ROW FORMAT SERDE
  'org.openx.data.jsonserde.JsonSerDe'
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/AWSLogs/accountID/WAFLogs/region/DOC-EXAMPLE-WEBACL/'
TBLPROPERTIES(
 'projection.enabled' = 'true',
 'projection.region.type' = 'enum',
 'projection.region.values' = 'us-east-1,us-west-2,eu-central-1,eu-west-1',
 'projection.date.type' = 'date',
 'projection.date.range' = '2021/01/01,NOW',
 'projection.date.format' = 'yyyy/MM/dd',
 'projection.date.interval' = '1',
 'projection.date.interval.unit' = 'DAYS',
 'storage.location.template' = 's3://DOC-EXAMPLE-BUCKET/AWSLogs/accountID/WAFLogs/
${region}/DOC-EXAMPLE-WEBACL/${date}/')
```

Note

Le format du chemin indiqué dans la `LOCATION` clause de l'exemple est standard mais peut varier en fonction de la AWS WAF configuration que vous avez implémentée. Par exemple, l'exemple de chemin de AWS WAF journal suivant concerne une CloudFront distribution :

```
s3://DOC-EXAMPLE-BUCKET/AWSLogs/12345678910/WAFLogs/cloudfront/
cloudfronyt/2022/08/08/17/55/
```

Si vous rencontrez des problèmes lors de la création ou de l'interrogation de votre table de AWS WAF journaux, confirmez l'emplacement de vos données de journal ou de votre contact AWS Support.

Pour plus d'informations sur la projection de partition, voir Projection de partition avec Amazon Athena.

Création d'une table pour les AWS WAF journaux sans partitionnement

Cette section explique comment créer une table pour les AWS WAF journaux sans partitionnement ni projection de partition.

> **Note**
>
> Pour des raisons de performance et de coût, nous vous déconseillons d'utiliser un schéma non partitionné pour les requêtes. Pour plus d'informations, consultez les 10 meilleurs conseils d'optimisation des performances pour Amazon Athena sur le blog AWS Big Data.

Pour créer le AWS WAF tableau

1. Copiez et collez l'instruction DDL suivante dans la console Athena. Modifiez les champs si nécessaire pour qu'ils correspondent à la sortie de votre journal. Modifiez l'emplacement `LOCATION` pour le compartiment Amazon S3 correspondant à celui qui stocke vos journaux.

   Cette requête utilise le OpenX JSON SerDe.

   > **Note**
   >
   > Il SerDe s'attend à ce que chaque document JSON se trouve sur une seule ligne de texte sans aucun caractère de fin de ligne séparant les champs de l'enregistrement. Si le texte JSON est dans un joli format d'impression, vous pouvez recevoir un message d'erreur tel que HIVE_CURSOR_ERROR : Row is not a valid JSON Object ou HIVE_CURSOR_ERROR : : Unexpected JsonParseException end-of-input : expected close marker for OBJECT lorsque vous essayez d'interroger la table après l'avoir créée. Pour plus d'informations, consultez la section Fichiers de données JSON dans la SerDe documentation OpenX sur. GitHub

```
CREATE EXTERNAL TABLE `waf_logs`(
  `timestamp` bigint,
  `formatversion` int,
  `webaclid` string,
  `terminatingruleid` string,
  `terminatingruletype` string,
  `action` string,
  `terminatingrulematchdetails` array <
                                  struct <
                                      conditiontype: string,
                                      sensitivitylevel: string,
                                      location: string,
```

```
                                        matcheddata: array < string >
                                          >
                                      >,
 `httpsourcename` string,
 `httpsourceid` string,
 `rulegrouplist` array <
                      struct <
                          rulegroupid: string,
                          terminatingrule: struct <
                                              ruleid: string,
                                              action: string,
                                              rulematchdetails: array <
                                                                  struct <
                                                                      conditiontype:
string,

sensitivitylevel: string,
                                                                      location:
string,
                                                                      matcheddata:
array < string >
                                                                         >
                                                                   >
                                            >,
                          nonterminatingmatchingrules: array <
                                                                struct <
                                                                    ruleid: string,
                                                                    action: string,
                                                                    overriddenaction:
string,
                                                                    rulematchdetails:
array <

 struct <

     conditiontype: string,

     sensitivitylevel: string,

     location: string,

     matcheddata: array < string >

        >
```

```
                                                                    >,
                                                                   challengeresponse:
struct <

responsecode: string,

solvetimestamp: string
                                                                          >,
                                                                   captcharesponse:
struct <

responsecode: string,

solvetimestamp: string
                                                                          >
                                                                    >
                                                                >,
                        excludedrules: string
                          >
                      >,
`ratebasedrulelist` array <
                        struct <
                            ratebasedruleid: string,
                            limitkey: string,
                            maxrateallowed: int
                              >
                         >,
 `nonterminatingmatchingrules` array <
                                  struct <
                                      ruleid: string,
                                      action: string,
                                      rulematchdetails: array <
                                                          struct <
                                                              conditiontype:
string,
                                                              sensitivitylevel:
string,
                                                              location: string,
                                                              matcheddata: array <
string >
                                                                 >
                                                          >,
                                      challengeresponse: struct <
                                                           responsecode: string,
```

```
                                                    solvetimestamp: string
                                                     >,
                                        captcharesponse: struct <
                                                    responsecode: string,
                                                    solvetimestamp: string
                                                     >
                                            >
                                    >,
`requestheadersinserted` array <
                                struct <
                                    name: string,
                                    value: string
                                      >
                                 >,
`responsecodesent` string,
`httprequest` struct <
                    clientip: string,
                    country: string,
                    headers: array <
                                struct <
                                    name: string,
                                    value: string
                                      >
                                 >,
                    uri: string,
                    args: string,
                    httpversion: string,
                    httpmethod: string,
                    requestid: string
                      >,
`labels` array <
                struct <
                    name: string
                      >
                 >,
`captcharesponse` struct <
                        responsecode: string,
                        solvetimestamp: string,
                        failureReason: string
                          >,
`challengeresponse` struct <
                        responsecode: string,
                        solvetimestamp: string,
                        failureReason: string
```

```
                        >,
  `ja3Fingerprint` string,
  `oversizefields` string,
  `requestbodysize` int,
  `requestbodysizeinspectedbywaf` int
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/'
```

2. Exécutez l'instruction `CREATE EXTERNAL TABLE` dans l'éditeur de requête de la console Athena. Cette opération permet d'enregistrer la table `waf_logs` et de mettre les données qu'elle contient à la disposition des requêtes d'Athena.

Exemples de requêtes pour les AWS WAF journaux

Bon nombre des exemples de requêtes suivants utilisent la table de projection de partition créée précédemment dans ce document. Dans les exemples de requêtes suivants, modifiez le nom de la table, les valeurs des colonnes et les autres variables en fonction de vos besoins. Pour augmenter les performances de vos requêtes et réduire les coûts, ajoutez la colonne de partition dans la condition de filtre.

- Count the number of referrers that contain a specified term
- Count all matched IP addresses in the last 10 days that have matched excluded rules
- Group all counted managed rules by the number of times matched
- Group all counted custom rules by number of times matched

Utilisation de la date et de l'heure

- Return the timestamp field in human-readable ISO 8601 format
- Return records from the last 24 hours
- Return records for a specified date range and IP address
- For a specified date range, count the number of IP addresses in five minute intervals
- Count the number of X-Forwarded-For IP in the last 10 days

Utilisation des requêtes et adresses bloquées

- Extract the top 100 IP addresses blocked by a specified rule type
- Count the number of times a request from a specified country has been blocked
- Count the number of times a request has been blocked, grouping by specific attributes
- Count the number of times a specific terminating rule ID has been matched
- Retrieve the top 100 IP addresses blocked during a specified date range

Example – Comptage du nombre de référents qui contiennent un terme spécifié

La requête suivante compte le nombre de référents qui contiennent le terme « amazon » pour la plage de dates spécifiée.

```
WITH test_dataset AS
  (SELECT header FROM waf_logs
    CROSS JOIN UNNEST(httprequest.headers) AS t(header) WHERE "date" >= '2021/03/01'
    AND "date" < '2021/03/31')
SELECT COUNT(*) referer_count
FROM test_dataset
WHERE LOWER(header.name)='referer' AND header.value LIKE '%amazon%'
```

Example – Comptage de toutes les adresses IP correspondant à des règles d'exclusion au cours des 10 derniers jours

La requête suivante compte le nombre de fois, au cours des 10 derniers jours, où l'adresse IP correspond à la règle d'exclusion dans le groupe de règles.

```
WITH test_dataset AS
  (SELECT * FROM waf_logs
    CROSS JOIN UNNEST(rulegrouplist) AS t(allrulegroups))
SELECT
  COUNT(*) AS count,
  "httprequest"."clientip",
  "allrulegroups"."excludedrules",
  "allrulegroups"."ruleGroupId"
```

```
FROM test_dataset
WHERE allrulegroups.excludedrules IS NOT NULL AND from_unixtime(timestamp/1000) > now()
 - interval '10' day
GROUP BY "httprequest"."clientip", "allrulegroups"."ruleGroupId",
 "allrulegroups"."excludedrules"
ORDER BY count DESC
```

Example – Regroupement de toutes les règles gérées comptées en fonction du nombre de fois qu'elles ont été mises en correspondance

Si vous avez défini les actions de règles du groupe de règles sur Count dans votre configuration ACL Web avant le 27 octobre 2022, vous AWS WAF avez enregistré vos remplacements dans le fichier JSON de l'ACL Web sous `excludedRules` le nom de. Désormais, le paramètre JSON permettant de remplacer une règle par Comptage se trouve dans les paramètres `ruleActionOverrides`. Pour plus d'informations, consultez Remplacements d'action dans les groupes de règles du Guide du développeur AWS WAF . Pour extraire les règles gérées en mode Comptage à partir de la nouvelle structure du journal, interrogez les `nonTerminatingMatchingRules` dans la section `ruleGroupList` plutôt que le champ `excludedRules`, comme dans l'exemple suivant.

```
SELECT
 count(*) AS count,
 httpsourceid,
 httprequest.clientip,
 t.rulegroupid,
 t.nonTerminatingMatchingRules
FROM "waf_logs"
CROSS JOIN UNNEST(rulegrouplist) AS t(t)
WHERE action <> 'BLOCK' AND cardinality(t.nonTerminatingMatchingRules) > 0
GROUP BY t.nonTerminatingMatchingRules, action, httpsourceid, httprequest.clientip,
 t.rulegroupid
ORDER BY "count" DESC
Limit 50
```

Example – Regroupement de toutes les règles personnalisées comptées en fonction du nombre de fois qu'elles ont été mises en correspondance

La requête suivante regroupe toutes les règles personnalisées comptées en fonction du nombre de fois qu'elles ont été mises en correspondance.

```
SELECT
```

```
  count(*) AS count,
         httpsourceid,
         httprequest.clientip,
         t.ruleid,
         t.action
FROM "waf_logs"
CROSS JOIN UNNEST(nonterminatingmatchingrules) AS t(t)
WHERE action <> 'BLOCK' AND cardinality(nonTerminatingMatchingRules) > 0
GROUP BY t.ruleid, t.action, httpsourceid, httprequest.clientip
ORDER BY "count" DESC
Limit 50
```

Pour plus d'informations sur l'emplacement des journaux pour les règles personnalisées et les groupes de règles gérés, consultez Surveillance et réglage du Guide du développeur AWS WAF .

Utilisation de la date et de l'heure

Example – Renvoi du champ d'horodatage au format ISO 8601 lisible par l'homme

La requête suivante utilise les fonctions `from_unixtime` et `to_iso8601` pour renvoyer le champ `timestamp` dans un format ISO 8601 lisible par l'homme (par exemple, `2019-12-13T23:40:12.000Z` au lieu de `1576280412771`). La requête renvoie également le nom de la source HTTP, l'ID de la source et la requête.

```
SELECT to_iso8601(from_unixtime(timestamp / 1000)) as time_ISO_8601,
       httpsourcename,
       httpsourceid,
       httprequest
FROM waf_logs
LIMIT 10;
```

Example – Renvoi des enregistrements des dernières 24 heures

La requête suivante utilise un filtre dans la clause `WHERE` pour renvoyer les champs Nom de la source HTTP, ID de la source HTTP et Requête HTTP pour les registres des dernières 24 heures.

```
SELECT to_iso8601(from_unixtime(timestamp/1000)) AS time_ISO_8601,
       httpsourcename,
       httpsourceid,
       httprequest
FROM waf_logs
```

```
WHERE from_unixtime(timestamp/1000) > now() - interval '1' day
LIMIT 10;
```

Example – Renvoi des enregistrements pour une plage de dates et une adresse IP spécifiées

La requête suivante répertorie les registres dans une plage de dates spécifiée pour une adresse IP client spécifiée.

```
SELECT *
FROM waf_logs
WHERE httprequest.clientip='53.21.198.66' AND "date" >= '2021/03/01' AND "date" <
 '2021/03/31'
```

Example – Comptage du nombre d'adresses IP par intervalles de cinq minutes pour une plage de dates spécifiée

La requête suivante compte, pour une plage de dates particulière, le nombre d'adresses IP à intervalles de cinq minutes.

```
WITH test_dataset AS
  (SELECT
     format_datetime(from_unixtime((timestamp/1000) -
 ((minute(from_unixtime(timestamp / 1000))%5) * 60)),'yyyy-MM-dd HH:mm') AS
 five_minutes_ts,
     "httprequest"."clientip"
     FROM waf_logs
     WHERE "date" >= '2021/03/01' AND "date" < '2021/03/31')
SELECT five_minutes_ts,"clientip",count(*) ip_count
FROM test_dataset
GROUP BY five_minutes_ts,"clientip"
```

Example – Comptage du nombre d'adresses IP X-Forwarded-For au cours des 10 derniers jours

La requête suivante filtre les en-têtes de requête et compte le nombre d'adresses IP X-Forwarded-For au cours des 10 derniers jours.

```
WITH test_dataset AS
  (SELECT header
   FROM waf_logs
   CROSS JOIN UNNEST (httprequest.headers) AS t(header)
   WHERE from_unixtime("timestamp"/1000) > now() - interval '10' DAY)
```

```
SELECT header.value AS ip,
       count(*) AS COUNT
FROM test_dataset
WHERE header.name='X-Forwarded-For'
GROUP BY header.value
ORDER BY COUNT DESC
```

Pour plus d'informations sur les fonctions de date et d'heure, veuillez consulter la rubrique Date and time functions and operators dans la documentation Trino.

Utilisation des requêtes et adresses bloquées

Example – Extraction des 100 premières adresses IP bloquées par un type de règle spécifié

La requête suivante extrait et compte les 100 premières adresses IP qui ont été bloquées par la règle de terminaison `RATE_BASED` pendant la période spécifiée.

```
SELECT COUNT(httpRequest.clientIp) as count,
httpRequest.clientIp
FROM waf_logs
WHERE terminatingruletype='RATE_BASED' AND action='BLOCK' and "date" >= '2021/03/01'
AND "date" < '2021/03/31'
GROUP BY httpRequest.clientIp
ORDER BY count DESC
LIMIT 100
```

Example – Comptage du nombre de fois où une demande provenant d'un pays donné a été bloquée

La requête suivante comptabilise le nombre de fois où la demande est arrivée à partir d'une adresse IP irlandaise (IE) et a été bloquée par la règle de résiliation `RATE_BASED`.

```
SELECT
  COUNT(httpRequest.country) as count,
  httpRequest.country
FROM waf_logs
WHERE
  terminatingruletype='RATE_BASED' AND
  httpRequest.country='IE'
GROUP BY httpRequest.country
ORDER BY count
LIMIT 100;
```

Example – Comptage du nombre de fois où une demande a été bloquée, en regroupant par attributs spécifiques

La requête suivante compte le nombre de fois où la demande a été bloquée, les résultats étant regroupés par WebACL, RuleId ClientIP et URI de requête HTTP.

```
SELECT
  COUNT(*) AS count,
  webaclid,
  terminatingruleid,
  httprequest.clientip,
  httprequest.uri
FROM waf_logs
WHERE action='BLOCK'
GROUP BY webaclid, terminatingruleid, httprequest.clientip, httprequest.uri
ORDER BY count DESC
LIMIT 100;
```

Example – Comptage du nombre de fois qu'un ID de règle de terminaison spécifique a été trouvé

La requête suivante comptabilise le nombre de fois où un ID de règle de résiliation spécifique a été mis en correspondance (`WHERE terminatingruleid='e9dd190d-7a43-4c06-bcea-409613d9506e'`). La requête regroupe ensuite les résultats par WebACL, Action, ClientIP et URI de requête HTTP.

```
SELECT
  COUNT(*) AS count,
  webaclid,
  action,
  httprequest.clientip,
  httprequest.uri
FROM waf_logs
WHERE terminatingruleid='e9dd190d-7a43-4c06-bcea-409613d9506e'
GROUP BY webaclid, action, httprequest.clientip, httprequest.uri
ORDER BY count DESC
LIMIT 100;
```

Example – Récupération des 100 premières adresses IP bloquées pendant une plage de dates spécifiée

La requête suivante extrait les 100 premières adresses IP qui ont été bloquées pour une plage de dates spécifiée. La requête répertorie également le nombre de fois où les adresses IP ont été bloquées.

```
SELECT "httprequest"."clientip", "count"(*) "ipcount", "httprequest"."country"
FROM waf_logs
WHERE "action" = 'BLOCK' and "date" >= '2021/03/01'
AND "date" < '2021/03/31'
GROUP BY "httprequest"."clientip", "httprequest"."country"
ORDER BY "ipcount" DESC limit 100
```

Pour plus d'informations sur l'interrogation des journaux de Simple Storage Service (Amazon S3), consultez les rubriques suivantes :

- Comment analyser mes journaux d'accès au serveur Simple Storage Service (Amazon S3) avec Athena ? dans le Centre de connaissances AWS
- Interrogation des journaux d'accès Amazon S3 pour les requêtes utilisant Amazon Athena dans le Guide de l'utilisateur Amazon Simple Storage Service
- Utilisation de AWS CloudTrail pour identifier les requêtes Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service

Interrogation des journaux de serveur web stockés dans Simple Storage Service (Amazon S3)

Vous pouvez utiliser Athena pour interroger les journaux de serveur web stockés dans Simple Storage Service (Amazon S3). Les rubriques de cette section vous montrent comment créer des tables dans Athena pour interroger des journaux de serveur web dans divers formats.

Rubriques

- Interrogation des journaux Apache stockés dans Simple Storage Service (Amazon S3)
- Interrogation de journaux Internet Information Server (IIS) stockés dans Simple Storage Service (Amazon S3)

Interrogation des journaux Apache stockés dans Simple Storage Service (Amazon S3)

Vous pouvez l'utiliser Amazon Athena pour interroger les fichiers journaux du serveur HTTP Apache stockés dans votre compte Amazon S3. Cette rubrique vous montre comment créer des schémas de table pour interroger les fichiers journaux d'accès Apache au format de journal commun.

Les champs du format de journal commun comprennent l'adresse IP du client, l'ID du client, l'ID de l'utilisateur, l'horodatage de la demande reçue, le texte de la demande du client, le code d'état du serveur et la taille de l'objet renvoyé au client.

L'exemple de données suivant montre le format de journal commun d'Apache.

```
198.51.100.7 - Li [10/Oct/2019:13:55:36 -0700] "GET /logo.gif HTTP/1.0" 200 232
198.51.100.14 - Jorge [24/Nov/2019:10:49:52 -0700] "GET /index.html HTTP/1.1" 200 2165
198.51.100.22 - Mateo [27/Dec/2019:11:38:12 -0700] "GET /about.html HTTP/1.1" 200 1287
198.51.100.9 - Nikki [11/Jan/2020:11:40:11 -0700] "GET /image.png HTTP/1.1" 404 230
198.51.100.2 - Ana [15/Feb/2019:10:12:22 -0700] "GET /favicon.ico HTTP/1.1" 404 30
198.51.100.13 - Saanvi [14/Mar/2019:11:40:33 -0700] "GET /intro.html HTTP/1.1" 200 1608
198.51.100.11 - Xiulan [22/Apr/2019:10:51:34 -0700] "GET /group/index.html HTTP/1.1"
 200 1344
```

Création d'une table dans Athena pour les journaux Apache

Avant de pouvoir interroger les journaux Apache stockés dans Simple Storage Service (Amazon S3), vous devez créer un schéma de table pour Athena afin qu'il puisse lire les données des journaux. Pour créer une table Athena pour les journaux Apache, vous pouvez utiliser le SerDe Grok SerDe. Pour plus d'informations sur l'utilisation du Grok SerDe, consultez la section Écrire des classificateurs personnalisés Grok dans le Guide du AWS Glue développeur.

Création d'une table dans Athena pour les journaux du serveur web Apache

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Collez l'instruction DDL suivante dans l'éditeur de requête Athena. Modifiez les valeurs dans `LOCATION 's3://DOC-EXAMPLE-BUCKET/apache-log-folder/'` pour pointer vers vos journaux Apache dans Simple Storage Service (Amazon S3).

   ```
   CREATE EXTERNAL TABLE apache_logs (
     client_ip string,
     client_id string,
     user_id string,
   ```

```
  request_received_time string,
  client_request string,
  server_status string,
  returned_obj_size string
  )
ROW FORMAT SERDE
   'com.amazonaws.glue.serde.GrokSerDe'
WITH SERDEPROPERTIES (
   'input.format'='^%{IPV4:client_ip} %{DATA:client_id} %{USERNAME:user_id}
 %{GREEDYDATA:request_received_time} %{QUOTEDSTRING:client_request}
 %{DATA:server_status} %{DATA: returned_obj_size}$'
   )
STORED AS INPUTFORMAT
   'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
   'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
   's3://DOC-EXAMPLE-BUCKET/apache-log-folder/';
```

3. Exécutez la requête dans la console Athena pour enregistrer la table `apache_logs`. Une fois la requête terminée, vous pouvez interroger les journaux à partir d'Athena.

Exemple de sélection des requêtes pour les journaux Apache

Example – Filtrage des erreurs 404

L'exemple de requête suivant sélectionne l'heure de réception de la demande, le texte de la demande du client et le code d'état du serveur à partir de la table `apache_logs`. La clause `WHERE` filtre le code d'état HTTP `404` (page non trouvée).

```
SELECT request_received_time, client_request, server_status
FROM apache_logs
WHERE server_status = '404'
```

L'image suivante montre les résultats de la requête dans l'éditeur de requête Athena.

Results

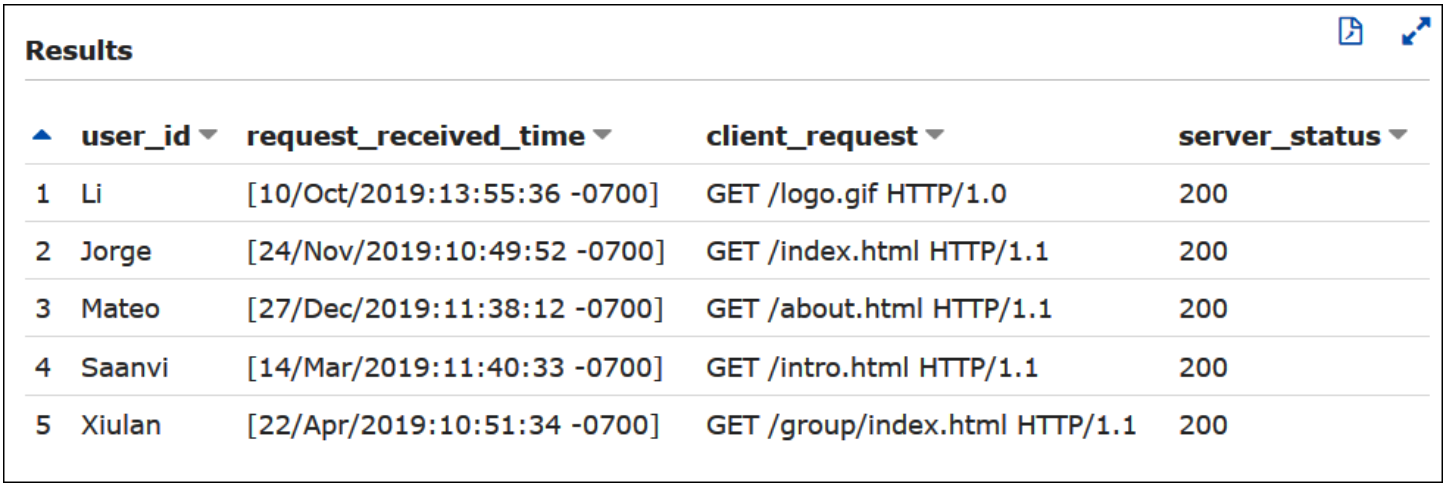
	request_received_time	client_request	server_status
1	[11/Jan/2020:11:40:11 -0700]	GET /image.png HTTP/1.1	404
2	[15/Feb/2019:10:12:22 -0700]	GET /favicon.ico HTTP/1.1	404

Example – Filtrage des demandes réussies

L'exemple de requête suivant sélectionne l'ID de l'utilisateur, l'heure de réception de la demande, le texte de la demande du client et le code d'état du serveur à partir de la table `apache_logs`. La clause `WHERE` filtre le code d'état HTTP `200` (réussite).

```
SELECT user_id, request_received_time, client_request, server_status
FROM apache_logs
WHERE server_status = '200'
```

L'image suivante montre les résultats de la requête dans l'éditeur de requête Athena.



Results

	user_id	request_received_time	client_request	server_status
1	Li	[10/Oct/2019:13:55:36 -0700]	GET /logo.gif HTTP/1.0	200
2	Jorge	[24/Nov/2019:10:49:52 -0700]	GET /index.html HTTP/1.1	200
3	Mateo	[27/Dec/2019:11:38:12 -0700]	GET /about.html HTTP/1.1	200
4	Saanvi	[14/Mar/2019:11:40:33 -0700]	GET /intro.html HTTP/1.1	200
5	Xiulan	[22/Apr/2019:10:51:34 -0700]	GET /group/index.html HTTP/1.1	200

Example – Filtrage par timestamp

L'exemple suivant demande des enregistrements dont l'heure de réception de la demande est supérieure à l'horodatage spécifié.

```
SELECT * FROM apache_logs WHERE request_received_time > 10/Oct/2023:00:00:00
```

Interrogation de journaux Internet Information Server (IIS) stockés dans Simple Storage Service (Amazon S3)

Vous pouvez utiliser Amazon Athena pour interroger les journaux des serveurs web de Microsoft Internet Information Services (IIS) stockés dans votre compte Simple Storage Service (Amazon S3). Bien que IIS utilise différents formats de fichiers journaux, cette rubrique vous montre comment créer des schémas de table pour interroger les journaux des formats de fichiers journaux étendus W3C et IIS à partir d'Athena.

Étant donné que les formats de fichier W3C Extended et IIS utilisent des séparateurs à caractère unique (espaces et virgules, respectivement) et que les valeurs ne sont pas placées entre guillemets, vous pouvez utiliser le LazySimpleSerDepour créer des tables Athena pour ces formats.

Utilisation du format de fichier journal étendu W3C

Le format de données du fichier journal étendu W3C comporte des champs séparés par des espaces. Les champs qui apparaissent dans les journaux étendus W3C sont déterminés par l'administrateur du serveur web qui choisit les champs à inclure dans les journaux. L'exemple suivant de données de journal comporte les champs `date, time`, `c-ip`, `s-ip`, `cs-method`, `cs-uri-stem`, `sc-status`, `sc-bytes`, `cs-bytes`, `time-taken` et `cs-version`.

```
2020-01-19 22:48:39 203.0.113.5 198.51.100.2 GET /default.html 200 540 524 157 HTTP/1.0
2020-01-19 22:49:40 203.0.113.10 198.51.100.12 GET /index.html 200 420 324 164 HTTP/1.0
2020-01-19 22:50:12 203.0.113.12 198.51.100.4 GET /image.gif 200 324 320 358 HTTP/1.0
2020-01-19 22:51:44 203.0.113.15 198.51.100.16 GET /faq.html 200 330 324 288 HTTP/1.0
```

Création d'une table dans Athena pour les journaux étendus W3C

Avant de pouvoir interroger vos journaux étendus W3C, vous devez créer un schéma de table pour qu'Athena puisse lire les données des journaux.

Créer une table dans Athena pour les journaux étendus W3C

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Collez une instruction DDL comme la suivante dans la console Athena, en tenant compte des points suivants :

    a. Ajoutez ou supprimez les colonnes dans l'exemple pour qu'elles correspondent aux champs des journaux que vous voulez interroger.

b. Les noms de colonne dans le format de fichier journal étendu W3C contiennent des traits d'union (-). Toutefois, conformément aux conventions de dénomination Athena, l'instruction `CREATE TABLE` en exemple les remplace par des traits de soulignement (_).

c. Pour spécifier le délimiteur espace, utilisez `FIELDS TERMINATED BY ' '`.

d. Modifiez les valeurs dans `LOCATION 's3://DOC-EXAMPLE-BUCKET/w3c-log-folder/'` pour pointer vers vos journaux étendus W3C dans Simple Storage Service (Amazon S3).

```
CREATE EXTERNAL TABLE `iis_w3c_logs`(
  date_col string,
  time_col string,
  c_ip string,
  s_ip string,
  cs_method string,
  cs_uri_stem string,
  sc_status string,
  sc_bytes string,
  cs_bytes string,
  time_taken string,
  cs_version 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://DOC-EXAMPLE-BUCKET/w3c-log-folder/'
```

3. Exécutez la requête dans la console Athena pour enregistrer la table `iis_w3c_logs`. Une fois la requête terminée, vous pouvez interroger les journaux à partir d'Athena.

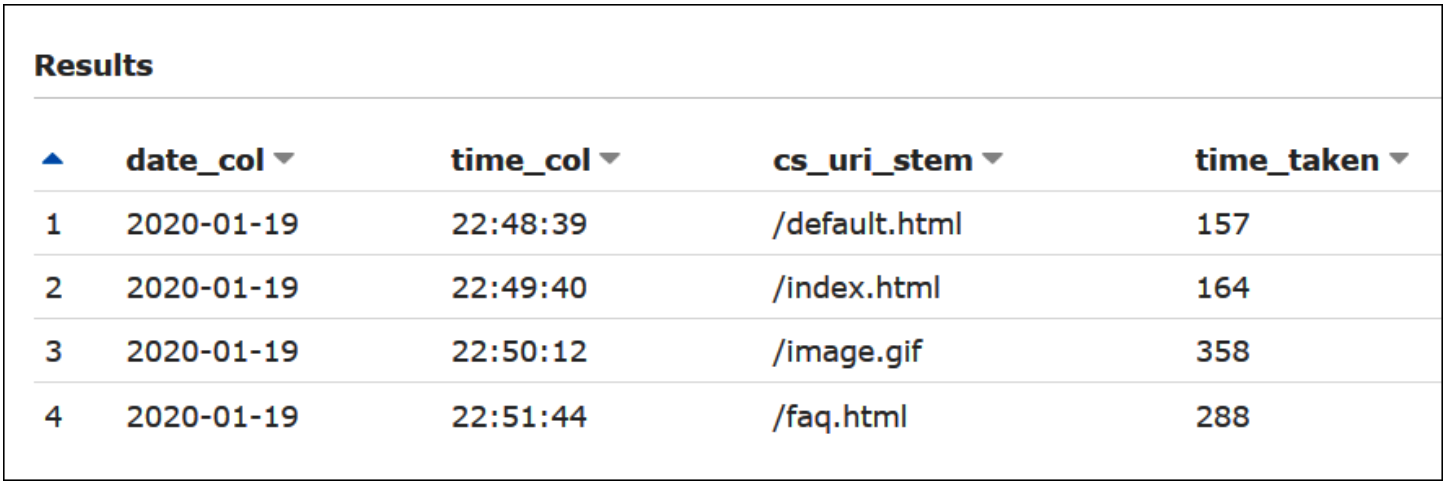
Exemple de requête de sélection de journal étendu W3C

L'exemple de requête suivant sélectionne la date, l'heure, la cible de la demande et le temps pris pour la demande à partir de la table `iis_w3c_logs`. La clause `WHERE` filtre les cas dans lesquels la méthode HTTP est `GET` et le code d'état HTTP est `200` (réussite).

```
SELECT date_col, time_col, cs_uri_stem, time_taken
```

```
FROM iis_w3c_logs
WHERE cs_method = 'GET' AND sc_status = '200'
```

L'image suivante montre les résultats de la requête dans l'éditeur de requête Athena.



Results

	date_col	time_col	cs_uri_stem	time_taken
1	2020-01-19	22:48:39	/default.html	157
2	2020-01-19	22:49:40	/index.html	164
3	2020-01-19	22:50:12	/image.gif	358
4	2020-01-19	22:51:44	/faq.html	288

Combination des champs de date et d'heure

Les champs `date` et `time` délimités par des espaces sont des entrées distinctes dans les données de la source du journal, mais vous pouvez les combiner en un horodatage si vous le souhaitez. Utilisez les fonctions concat() et date_parse() dans une requête SELECT ou CREATE TABLE AS SELECT pour concaténer et convertir les colonnes de date et d'heure au format horodatage. L'exemple suivant utilise une requête CTAS pour créer une nouvelle table avec une colonne `derived_timestamp`.

```
CREATE TABLE iis_w3c_logs_w_timestamp AS
SELECT
  date_parse(concat(date_col,' ', time_col),'%Y-%m-%d %H:%i:%s') as derived_timestamp,
  c_ip,
  s_ip,
  cs_method,
  cs_uri_stem,
  sc_status,
  sc_bytes,
  cs_bytes,
  time_taken,
  cs_version
FROM iis_w3c_logs
```

Une fois la table créée, vous pouvez interroger directement la nouvelle colonne d'horodatage, comme dans l'exemple suivant.

```
SELECT derived_timestamp, cs_uri_stem, time_taken
FROM iis_w3c_logs_w_timestamp
WHERE cs_method = 'GET' AND sc_status = '200'
```

L'image suivante montre les résultats de la requête.

Results

	derived_timestamp	cs_uri_stem	time_taken
1	2020-01-19 22:48:39.000	/default.html	157
2	2020-01-19 22:49:40.000	/index.html	164
3	2020-01-19 22:50:12.000	/image.gif	358
4	2020-01-19 22:51:44.000	/faq.html	288

Format de fichier journal IIS

Contrairement au format étendu W3C, le format de fichier journal IIS comporte un ensemble fixe de champs et inclut une virgule comme délimiteur. LazySimpleSerDe Traite la virgule comme le délimiteur et l'espace après la virgule comme le début du champ suivant.

L'exemple suivant montre des données types dans le format de fichier journal IIS.

```
203.0.113.15, -, 2020-02-24, 22:48:38, W3SVC2, SERVER5, 198.51.100.4, 254, 501, 488,
 200, 0, GET, /index.htm, -,
203.0.113.4, -, 2020-02-24, 22:48:39, W3SVC2, SERVER6, 198.51.100.6, 147, 411, 388,
 200, 0, GET, /about.html, -,
203.0.113.11, -, 2020-02-24, 22:48:40, W3SVC2, SERVER7, 198.51.100.18, 170, 531, 468,
 200, 0, GET, /image.png, -,
203.0.113.8, -, 2020-02-24, 22:48:41, W3SVC2, SERVER8, 198.51.100.14, 125, 711, 868,
 200, 0, GET, /intro.htm, -,
```

Création d'une table dans Athena pour les fichiers journaux IIS

Pour interroger les journaux au format de fichier journal IIS dans Simple Storage Service (Amazon S3), vous devez d'abord créer un schéma de table afin qu'Athena puisse lire les données du journal.

Création d'une table dans Athena pour les journaux au format de fichier journal IIS

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.

2. Collez l'instruction DDL suivante dans la console Athena, en tenant compte des points suivants :

   a. Pour spécifier le délimiteur virgule, utilisez `FIELDS TERMINATED BY ','`.

   b. Modifiez les valeurs dans LOCATION 's3://DOC-EXAMPLE-BUCKET/ *iis-log-file-folder*/' pour pointer vers vos fichiers journaux au format journal IIS dans Amazon S3.

```
CREATE EXTERNAL TABLE `iis_format_logs`(
client_ip_address string,
user_name string,
request_date string,
request_time string,
service_and_instance string,
server_name string,
server_ip_address string,
time_taken_millisec string,
client_bytes_sent string,
server_bytes_sent string,
service_status_code string,
windows_status_code string,
request_type string,
target_of_operation string,
script_parameters 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://DOC-EXAMPLE-BUCKET/iis-log-file-folder/'
```

3. Exécutez la requête dans la console Athena pour enregistrer la table `iis_format_logs`. Une fois la requête terminée, vous pouvez interroger les journaux à partir d'Athena.

Exemple de requête de sélection du format de journal IIS

L'exemple de requête suivant sélectionne la date de la demande, l'heure de la demande, la cible de la demande et le temps pris en millisecondes à partir de la table `iis_format_logs`. La clause `WHERE` filtre les cas dans lesquels le type de requête est `GET` et le code d'état HTTP est `200`

(réussite). Dans la requête, notez que les espaces de tête dans `' GET'` et `' 200'` sont nécessaires pour que la requête réussisse.

```
SELECT request_date, request_time, target_of_operation, time_taken_millisec
FROM iis_format_logs
WHERE request_type = ' GET' AND service_status_code = ' 200'
```

L'image suivante montre les résultats de la requête sur l'échantillon de données.

Results

	request_date	request_time	target_of_operation	time_taken_millisec
1	2020-02-24	22:48:38	/index.htm	254
2	2020-02-24	22:48:39	/about.html	147
3	2020-02-24	22:48:40	/image.png	170
4	2020-02-24	22:48:41	/intro.htm	125

Format de fichier journal NCSA

IIS utilise également le format de Journalisation NCSA, qui comporte un nombre fixe de champs au format texte ASCII séparés par des espaces. La structure est similaire au format de journal commun utilisé pour les journaux d'accès Apache. Les champs du format de données du journal commun NCSA comprennent l'adresse IP du client, l'ID du client (généralement non utilisé), l'ID de l'utilisateur du domaine, l'horodatage de la demande reçue, le texte de la demande du client, le code d'état du serveur et la taille de l'objet renvoyé au client.

L'exemple suivant montre des données au format de journal commun NCSA tel que documenté pour IIS.

```
198.51.100.7 - ExampleCorp\Li [10/Oct/2019:13:55:36 -0700] "GET /logo.gif HTTP/1.0" 200
 232
198.51.100.14 - AnyCompany\Jorge [24/Nov/2019:10:49:52 -0700] "GET /index.html
 HTTP/1.1" 200 2165
198.51.100.22 - ExampleCorp\Mateo [27/Dec/2019:11:38:12 -0700] "GET /about.html
 HTTP/1.1" 200 1287
198.51.100.9 - AnyCompany\Nikki [11/Jan/2020:11:40:11 -0700] "GET /image.png HTTP/1.1"
 404 230
198.51.100.2 - ExampleCorp\Ana [15/Feb/2019:10:12:22 -0700] "GET /favicon.ico HTTP/1.1"
 404 30
```

```
198.51.100.13 - AnyCompany\Saanvi [14/Mar/2019:11:40:33 -0700] "GET /intro.html
 HTTP/1.1" 200 1608
198.51.100.11 - ExampleCorp\Xiulan [22/Apr/2019:10:51:34 -0700] "GET /group/index.html
 HTTP/1.1" 200 1344
```

Création d'une table dans Athena pour les journaux NCSA IIS

Pour votre instruction `CREATE TABLE`, vous pouvez utiliser le SerDe Grok SerDe et un motif Grok similaire à celui utilisé pour les journaux du serveur web Apache. Contrairement aux journaux Apache, le motif Grok utilise `%{DATA:user_id}` pour le troisième champ au lieu de `%{USERNAME:user_id}` pour tenir compte de la présence de la barre oblique inverse dans `domain\user_id`. Pour plus d'informations sur l'utilisation du Grok SerDe, consultez la section Écrire des classificateurs personnalisés Grok dans le Guide du AWS Glue développeur.

Création d'une table dans Athena pour les journaux de serveur web NCSA IIS

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Collez l'instruction DDL suivante dans l'éditeur de requête Athena. Modifiez les valeurs dans `LOCATION 's3://DOC-EXAMPLE-BUCKET/iis-ncsa-logs/'` pour pointer vers vos journaux NCSA IIS dans Simple Storage Service (Amazon S3).

```
CREATE EXTERNAL TABLE iis_ncsa_logs(
  client_ip string,
  client_id string,
  user_id string,
  request_received_time string,
  client_request string,
  server_status string,
  returned_obj_size string
  )
ROW FORMAT SERDE
   'com.amazonaws.glue.serde.GrokSerDe'
WITH SERDEPROPERTIES (
   'input.format'='^%{IPV4:client_ip} %{DATA:client_id} %{DATA:user_id}
 %{GREEDYDATA:request_received_time} %{QUOTEDSTRING:client_request}
 %{DATA:server_status} %{DATA: returned_obj_size}$'
   )
STORED AS INPUTFORMAT
   'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
   'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
```

```
    's3://DOC-EXAMPLE-BUCKET/iis-ncsa-logs/';
```

3. Exécutez la requête dans la console Athena pour enregistrer la table `iis_ncsa_logs`. Une fois la requête terminée, vous pouvez interroger les journaux à partir d'Athena.

Exemple de sélection de requêtes pour les journaux IIS NCSA

Example – Filtrage des erreurs 404

L'exemple de requête suivant sélectionne l'heure de réception de la demande, le texte de la demande du client et le code d'état du serveur à partir de la table `iis_ncsa_logs`. La clause `WHERE` filtre le code d'état HTTP `404` (page non trouvée).

```
SELECT request_received_time, client_request, server_status
FROM iis_ncsa_logs
WHERE server_status = '404'
```

L'image suivante montre les résultats de la requête dans l'éditeur de requête Athena.

Results

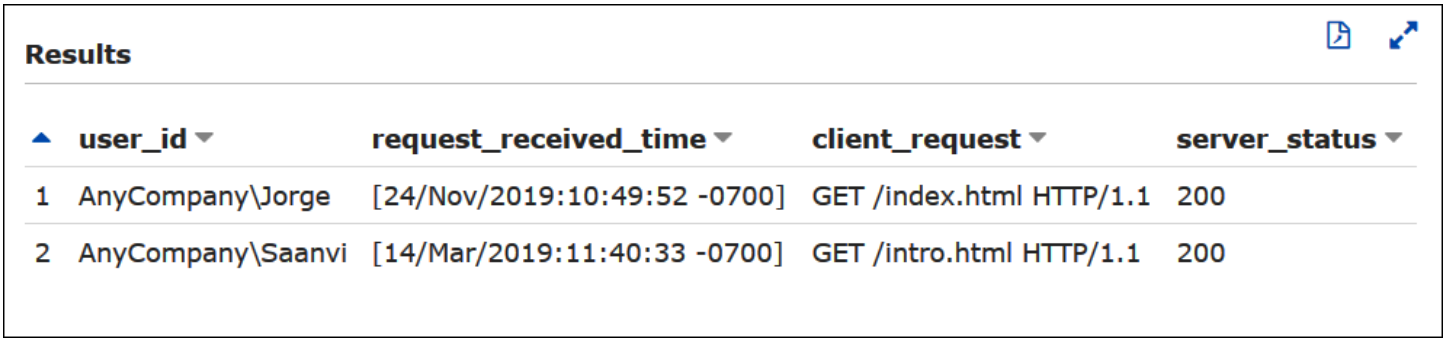
	request_received_time	client_request	server_status
1	[11/Jan/2020:11:40:11 -0700]	GET /image.png HTTP/1.1	404
2	[15/Feb/2019:10:12:22 -0700]	GET /favicon.ico HTTP/1.1	404

Example – Filtrage des requêtes réussies en provenance d'un domaine particulier

L'exemple de requête suivant sélectionne l'ID de l'utilisateur, l'heure de réception de la demande, le texte de la demande du client et le code d'état du serveur à partir de la table `iis_ncsa_logs`. La clause `WHERE` filtre les requêtes avec le code d'état HTTP `200` (réussite) provenant d'utilisateurs du domaine `AnyCompany`.

```
SELECT user_id, request_received_time, client_request, server_status
FROM iis_ncsa_logs
WHERE server_status = '200' AND user_id LIKE 'AnyCompany%'
```

L'image suivante montre les résultats de la requête dans l'éditeur de requête Athena.



Results

	user_id	request_received_time	client_request	server_status
1	AnyCompany\Jorge	[24/Nov/2019:10:49:52 -0700]	GET /index.html HTTP/1.1	200
2	AnyCompany\Saanvi	[14/Mar/2019:11:40:33 -0700]	GET /intro.html HTTP/1.1	200

Utilisation des transactions Athena ACID

Le terme « transactions ACID » fait référence à un ensemble de propriétés (atomicité, cohérence, isolation et durabilité) qui garantissent l'intégrité des données dans les transactions de bases de données. Les transactions ACID permettent à plusieurs utilisateurs d'ajouter et de supprimer simultanément et en toute fiabilité des objets Simple Storage Service (Amazon S3) de manière atomique, tout en isolant les requêtes existantes en maintenant la cohérence de lecture pour les requêtes contre le lac de données. Les transactions Athena ACID ajoutent au langage de manipulation des données (DML) Athena SQL la prise en charge d'une table unique pour les opérations d'insertion, de suppression, de mise à jour et Time Travel. Vous et plusieurs utilisateurs simultanés pouvez utiliser les transactions Athena ACID pour apporter des modifications fiables, au niveau des lignes, aux données Amazon S3. Les transactions Athena gèrent automatiquement la sémantique et la coordination du verrouillage et ne nécessitent pas de solution de verrouillage d'enregistrement personnalisée.

Les transactions Athena ACID et la syntaxe SQL familière simplifient les mises à jour de vos données commerciales et réglementaires. Par exemple, pour répondre à une demande d'effacement de données, vous pouvez effectuer une opération SQL `DELETE`. Pour effectuer des corrections d'enregistrement manuelles, vous pouvez utiliser une seule instruction `UPDATE`. Pour récupérer des données qui ont été récemment supprimées, vous pouvez émettre des requêtes Time Travel en utilisant une instruction `SELECT`.

Parce qu'elles sont basées sur des formats de tables partagées, les transactions Athena ACID sont compatibles avec d'autres services et moteurs tels que Amazon EMR et Apache Spark qui prennent également en charge les formats de tables partagés.

Les transactions Athena sont disponibles via la console Athena, les opérations API et les pilotes ODBC et JDBC.

Rubriques

Interrogation des tables Linux Foundation Delta Lake

Linux Foundation Delta Lake est un format de table pour l'analytique du big data. Vous pouvez utiliser Amazon Athena pour lire directement les tables Delta Lake stockées dans Amazon S3 sans avoir à générer de fichiers manifestes ou à exécuter l'instruction `MSCK REPAIR`.

Le format Delta Lake stocke les valeurs minimales et maximales par colonne de chaque fichier de données. L'implémentation Athena utilise ces informations pour permettre le saut de fichier sur les prédicats afin d'éliminer les fichiers non désirés de la considération.

Considérations et restrictions

L'assistance de Delta Lake à Athena comporte les considérations et limites suivantes :

- Tableaux avec AWS Glue catalogue uniquement : le support natif de Delta Lake n'est pris en charge que par le biais des tables enregistrées auprès de AWS Glue. Si une table Delta Lake est enregistrée dans un autre métastore, vous pouvez la conserver et la traiter comme votre métastore primaire. Les métadonnées de Delta Lake étant stockées dans le système de fichiers (par exemple, dans Amazon S3) plutôt que dans le métastore, Athena n'a besoin que de la propriété location in AWS Glue pour lire vos tables Delta Lake.
- Moteur V3 uniquement – Les requêtes Delta Lake ne sont prises en charge que par la version 3 du moteur Athena. Vous devez vous assurer que le groupe de travail que vous créez est configuré pour utiliser la version 3 du moteur Athena.
- Version du lecteur Delta Lake — Le protocole de lecteur Delta Lake jusqu'à la version 3 est pris en charge.
- Mappage des colonnes et timestampNTZ : le mappage des colonnes Delta, qui permet aux colonnes des tables Delta et aux colonnes du fichier Parquet sous-jacent d'utiliser des noms différents, et l'horodatage sans fuseau horaire (TimestampNTZ) sont pris en charge.
- Pas de prise en charge du voyage dans le temps – Il n'y a pas de prise en charge des requêtes qui utilisent les capacités de voyage dans le temps de Delta Lake.
- Lecture seule – L'écriture d'instructions DML comme `UPDATE`, `INSERT` ou `DELETE` n'est pas prise en charge.

- Prise en charge de Lake Formation : l'intégration de Lake Formation est disponible pour les tables Delta Lake dont le schéma est synchronisé avec AWS Glue. Pour plus d'informations, consultez les AWS Lake Formation sections Utilisation avec Amazon Athena et Configuration des autorisations pour une table Delta Lake dans le guide du AWS Lake Formation développeur.
- Prise en charge limitée de DDL – Les instructions DDL suivantes sont prises en charge : `CREATE EXTERNAL TABLE`, `SHOW COLUMNS`, `SHOW TBLPROPERTIES`, `SHOW PARTITIONS`, `SHOW CREATE TABLE` et `DESCRIBE`. Pour plus d'informations sur l'utilisation de l'instruction `CREATE EXTERNAL TABLE`, voir la rubrique Premiers pas.
- Omission d'objets S3 Glacier non prise en charge : si les objets de la table Delta Lake de Linux Foundation appartiennent à une classe de stockage Amazon S3 Glacier, la définition de la propriété de table `read_restored_glacier_objects` sur `false` n'a aucun effet.

  Par exemple, supposons que vous exécutiez la commande suivante :

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

  Pour les tables Iceberg et Delta Lake, la commande produit l'erreur Unsupported table property key: read_restored_glacier_objects. Pour les tables Hudi, la commande `ALTER TABLE` ne produit pas d'erreur, mais les objets Amazon S3 Glacier ne sont toujours pas ignorés. L'exécution de requêtes `SELECT` après la commande `ALTER TABLE` continue de renvoyer tous les objets.

Prise en charge des types de données de colonnes non-partition

Pour les colonnes non-partition, tous les types de données pris en charge par Athena, à l'exception de `CHAR`, sont pris en charge (`CHAR` n'est pas pris en charge par le protocole Delta Lake lui-même). Les types de données pris en charge sont les suivants :

```
boolean
tinyint
smallint
integer
bigint
double
float
decimal
varchar
string
binary
```

```
date
timestamp
array
map
struct
```

Prise en charge des types de données de colonnes partition

Pour les colonnes de type partition, Athena prend en charge les tables avec les types de données suivants :

```
boolean
integer
smallint
tinyint
bigint
decimal
float
double
date
timestamp
varchar
```

Pour de plus amples informations sur les types de données dans Athena, voir Types de données dans Amazon Athena.

Premiers pas

Pour pouvoir être interrogée, votre table Delta Lake doit exister dans. AWS Glue Si votre table se trouve dans Amazon S3 mais pas dans AWS Glue, exécutez une `CREATE EXTERNAL TABLE` instruction en utilisant la syntaxe suivante. Si votre table existe déjà dans AWS Glue (par exemple, parce que vous utilisez Apache Spark ou un autre moteur AWS Glue), vous pouvez ignorer cette étape.

```
CREATE EXTERNAL TABLE
  [db_name.]table_name
  LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
  TBLPROPERTIES ('table_type' = 'DELTA')
```

Notez l'omission des définitions de colonnes, de SerDe la bibliothèque et des autres propriétés de table. Contrairement aux tables Hive traditionnelles, les métadonnées des tables Delta Lake sont déduites du journal des transactions de Delta Lake et synchronisées directement avec. AWS Glue

> **Note**
>
> Pour les tables Delta Lake, les instructions `CREATE TABLE` qui comprennent plus que les propriétés `LOCATION` et `table_type` ne sont pas autorisées.

Lecture des tables Delta Lake

Pour interroger une table Delta Lake, utilisez la syntaxe SQL `SELECT` standard :

```
[ WITH with_query [, ...] ]SELECT [ ALL | DISTINCT ] select_expression [, ...]
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
[ HAVING condition ]
[ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select ]
[ ORDER BY expression [ ASC | DESC ] [ NULLS FIRST | NULLS LAST] [, ...] ]
[ OFFSET count [ ROW | ROWS ] ]
[ LIMIT [ count | ALL ] ]
```

Pour plus d'informations sur la syntaxe `SELECT`, voir SELECT dans la documentation d'Athena.

Le format Delta Lake stocke les valeurs minimales et maximales par colonne de chaque fichier de données. Athena utilise ces informations pour permettre le saut de fichiers sur les prédicats afin d'éliminer les fichiers inutiles de la considération.

Synchronisation des métadonnées Delta Lake

Athena synchronise les métadonnées des tables, y compris le schéma, les colonnes de partition et les propriétés des tables, AWS Glue si vous utilisez Athena pour créer votre table Delta Lake. Au fil du temps, ces métadonnées peuvent perdre leur synchronisation avec les métadonnées de la table sous-jacente dans le journal des transactions. Pour garder votre table à jour, vous pouvez choisir l'une des options suivantes :

- Utilisez le AWS Glue crawler pour les tables Delta Lake. Pour plus d'informations, consultez les sections Présentation de la prise en charge native des tables Delta Lake avec AWS Glue les robots

d'exploration dans le blog AWS Big Data et Planification d'un AWS Glue robot d'exploration dans le guide du AWS Glue développeur.

- Annulez et recréez la table dans Athena.
- Utilisez le SDK, la CLI ou AWS Glue la console pour mettre à jour manuellement le schéma dans AWS Glue.

Notez que les fonctionnalités suivantes nécessitent que votre AWS Glue schéma ait toujours le même schéma que le journal des transactions :

- Lake Formation
- Vues
- Filtres de lignes et de colonnes

Si votre flux de travail ne nécessite aucune de ces fonctionnalités et que vous préférez ne pas conserver cette compatibilité, vous pouvez utiliser le `CREATE TABLE` DDL dans Athena, puis ajouter le chemin Amazon S3 en SerDe tant que paramètre. AWS Glue

Pour créer une table Delta Lake à l'aide de l'Athena et des consoles AWS Glue

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans l'éditeur de requêtes Athena, utilisez l'instruction DDL suivante pour créer votre table Delta Lake. Notez que lorsque vous utilisez cette méthode, la valeur de `TBLPROPERTIES` doit être `'spark.sql.sources.provider' = 'delta'` et non `'table_type' = 'delta'`.

   Notez que ce même schéma (avec une seule colonne nommée `col` de type `array<string>`) est inséré lorsque vous utilisez Apache Spark (Athena pour Apache Spark) ou la plupart des autres moteurs pour créer votre table.

   ```
   CREATE EXTERNAL TABLE
      [db_name.]table_name(col array<string>)
      LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
      TBLPROPERTIES ('spark.sql.sources.provider' = 'delta')
   ```
3. Ouvrez la AWS Glue console à l'adresse https://console.aws.amazon.com/glue/.
4. Dans le panneau de navigation, choisissez Catalogue de données, Tables.
5. Dans la liste des tables, choisissez le lien correspondant à votre table.

6. Sur la page de la table, choisissez Actions, Modifier la table.
7. Dans la section Paramètres Serde, ajoutez la clé **`path`** avec la valeur **`s3://DOC-EXAMPLE-BUCKET/your-folder/`**.
8. Choisissez Save (Enregistrer).

Ressources supplémentaires

Pour une discussion sur l'utilisation des tables Delta Lake avec Athena AWS Glue et leur interrogation avec Athena, voir Gérer les opérations de données UPSERT à l'aide de l'open source Delta Lake AWS Glue et sur le blog Big Data.AWS

Utilisation d'Athena pour interroger des jeux de données Apache Hudi

Apache Hudi est un cadre de gestion de données open source qui simplifie le traitement progressif des données. Les actions d'insertion, de mise à jour, d'insertion ascendante et de suppression au niveau des registres sont traitées de manière beaucoup plus granulaire, ce qui réduit les frais généraux. `Upsert` désigne la possibilité d'insérer des registres dans un jeu de données existant s'ils n'existent pas encore ou de les mettre à jour s'ils existent déjà.

Hudi traite les événements d'insertion et de mise à jour des données sans créer de nombreux petits fichiers qui peuvent entraîner des problèmes de performance pour les analyses. Apache Hudi suit automatiquement les modifications et fusionne les fichiers afin qu'ils conservent une taille optimale. Cela évite de devoir créer des solutions personnalisées qui contrôlent et réécrivent de nombreux petits fichiers en un nombre réduit de gros fichiers.

Les jeux de données Hudi sont adaptés pour les cas d'utilisation suivants :

- Conformité aux réglementations relatives à la protection de la vie privée, comme le Règlement général sur la protection des données (RGPD) et La loi californienne sur la protection de la vie privée des consommateurs (CCPA), qui font respecter le droit des personnes à supprimer leurs informations personnelles ou à modifier la manière dont leurs données sont utilisées.
- Utilisation des données de streaming provenant de capteurs et d'autres appareils IoT (Internet des objets) nécessitant des événements d'insertion et de mise à jour spécifiques.
- Mise en œuvre d'un système de capture de données modifiées (CDC, Change Data Capture).

Les jeux de données gérés par Hudi sont stockés dans Amazon S3 en utilisant des formats de stockage ouverts. Actuellement, Athena peut lire des jeux de données Hudi compactés, mais pas

écrire des données Hudi. Athena prend en charge jusqu'à la version 0.8.0 de Hudi avec la version 2 du moteur Athena, et la version 0.14.0 de Hudi avec la version 3 du moteur Athena. Cela est susceptible de changer. Athena ne peut garantir la compatibilité de lecture avec les tables créées avec des versions ultérieures de Hudi. Pour plus d'informations sur la gestion des versions du moteur Athena, voir Gestion des versions du moteur Athena. Pour plus d'informations sur les fonctions et la gestion des versions de Hudi, voir la documentation Hudi sur le site Web Apache.

Types de table de jeux de données Hudi

Un jeu de données Hudi peut être l'un des types suivants :

- Copie sur écriture (CoW) – Les données sont stockées dans un format en colonnes (Parquet) et chaque mise à jour crée une nouvelle version des fichiers lors d'une écriture.
- Fusion sur lecture (MoR) – Les données sont stockées en utilisant une combinaison de formats en colonnes (Parquet) et en lignes (Avro). Les mises à jour sont enregistrées dans les fichiers `delta` en lignes et sont compactées si nécessaire pour créer de nouvelles versions des fichiers en colonnes.

Avec les ensembles de données CoW, chaque fois qu'une mise à jour est apportée à un enregistrement, le fichier qui contient l'enregistrement est réécrit avec les valeurs mises à jour. Avec un jeu de données MoR, chaque fois qu'une mise à jour a lieu, Hudi écrit uniquement la ligne du registre modifié. Le type de stockage MoR est mieux adapté aux charges de travail donnant lieu à de nombreuses écritures ou modifications avec moins de lectures. Le type de stockage CoW est mieux adapté aux charges de travail lourdes en lecture sur des données qui changent moins fréquemment.

Hudi propose trois types de requêtes pour accéder aux données :

- Requêtes d'instantané – Requêtes qui affichent le dernier instantané de la table à partir d'une action de validation ou de compactage donnée. Pour les tables MoR, les requêtes d'instantané exposent l'état le plus récent de la table en fusionnant les fichiers de base et delta de la dernière tranche de fichiers au moment de la requête.
- Requête progressives – Les requêtes ne portent que sur les nouvelles données écrites dans la table, depuis une validation/un compactage donné. Cela fournit efficacement des flux de modifications pour activer les pipelines de données (Data Pipelines) progressives.
- Requêtes à lecture optimisée – Pour les tables MoR, les requêtes portent sur les dernières données compactées. Pour les tables CoW, les requêtes portent sur les dernières données validées.

La table suivante montre les types de requêtes Hudi possibles pour chaque type de table.

Type de table	Types de requêtes Hudi possibles
Copie sur écriture	instantané, progressif
fusion sur lecture	instantané, progressif, lecture optimisée

Actuellement, Athena prend en charge les requêtes d'instantané et les requêtes à lecture optimisée, mais pas les requêtes progressives. Sur les tables MoR, toutes les données exposées aux requêtes à lecture optimisée sont compactées . Cela procure de bonnes performances, mais n'inclut pas les dernières validations delta. Les requêtes d'instantané contiennent les données les plus récentes mais entraînent une surcharge de calcul, ce qui rend ces requêtes moins performantes.

Pour plus d'informations sur les compromis entre les types de tables et de requêtes, consultez Types de tables et de requêtes dans la documentation d'Apache Hudi.

Changement de terminologie Hudi : les vues sont désormais des requêtes

À partir de la version 0.5.1, Apache Hudi a modifié une partie de sa terminologie. Les anciennes vues sont appelées requêtes dans les versions ultérieures. La table suivante résume les anciens et les nouveaux termes.

Ancien terme	Nouveau terme
CoW : vue à lecture optimisée MoR : vue en temps réel	Requêtes d'instantané
Vue progressive	Requête progressive
Vue à lecture optimisée MoR	Requête à lecture optimisée

Tables de l'opération d'amorçage

À partir de la version 0.6.0 d'Apache Hudi, la fonction d'opération d'amorçage offre de meilleures performances avec les jeux de données Parquet existants. Au lieu de réécrire le jeu de données, une opération d'amorçage ne peut générer que des métadonnées, laissant le jeu de données en place.

Vous pouvez utiliser Athena pour interroger des tables à partir d'une opération d'amorçage, comme d'autres tables basées sur des données dans Simple Storage Service (Amazon S3). Dans votre instruction `CREATE TABLE`, spécifiez le chemin d'accès de la table Hudi dans votre clause `LOCATION`.

Pour plus d'informations sur la création de tables Hudi à l'aide de l'opération bootstrap dans Amazon EMR, consultez l'article Nouvelles fonctionnalités d'Apache Hudi disponibles sur Amazon EMR sur le Big Data Blog. AWS

Listage des métadonnées Hudi

Apache Hudi possède une table de métadonnées qui contient des fonctionnalités d'indexation pour améliorer les performances, telles que le listage des fichiers, le saut de données à l'aide de statistiques de colonnes et un index basé sur un filtre Bloom.

Parmi ces fonctionnalités, Athena ne prend actuellement en charge que l'index de listage des fichiers. L'index de listage des fichiers élimine les appels au système de fichiers tels que les « list files » en récupérant les informations d'un index qui gère un mappage d'une partition à des fichiers. Cela élimine le besoin de répertorier de manière récursive chaque partition sous le chemin de la table pour obtenir une vue du système de fichiers. Lorsque vous travaillez avec de grands jeux de données, cette indexation réduit considérablement la latence qui se produirait sinon lors de l'obtention de la liste des fichiers durant les écritures et les requêtes. Cela permet également d'éviter les goulots d'étranglement tels que la limitation des limites de demandes lors des appels `LIST` Amazon S3.

> **Note**
>
> Athena ne prend actuellement pas en charge le saut de données ni l'indexation par filtre Bloom.

Activation de la table de métadonnées Hudi

Le listage des fichiers basé sur les tables de métadonnées est désactivée par défaut. Pour activer la table de métadonnées Hudi et la fonctionnalité de listage des fichiers associée, définissez la propriété de table `hudi.metadata-listing-enabled` sur `TRUE`.

Exemple

L'exemple `ALTER TABLE SET TBLPROPERTIES` suivant active la table de métadonnées dans la table `partition_cow` d'exemple.

```
ALTER TABLE partition_cow SET TBLPROPERTIES('hudi.metadata-listing-enabled'='TRUE')
```

Considérations et restrictions

- Athena ne prend pas en charge les requêtes progressives.
- Athena ne prend pas en charge CTAS ou INSERT INTO sur les données Hudi. Si vous souhaitez qu'Athena prenne en charge l'écriture des jeux de données Hudi, envoyez vos commentaires à `<athena-feedback@amazon.com>`.

  Pour plus d'informations sur l'écriture des données Hudi, consultez les ressources suivantes :

  - Utilisation d'un jeu de données Hudi dans le Guide de version Amazon EMR.
  - Écriture de données dans la documentation d'Apache Hudi.
- L'utilisation de MSCK REPARATION TABLE sur les tables Hudi dans Athena n'est pas prise en charge. Si vous devez charger une table Hudi qui n'a pas été créée dans AWS Glue, utilisezALTER TABLE ADD PARTITION.
- Omission d'objets S3 Glacier non prise en charge : si les objets de la table Apache Hudi appartiennent à une classe de stockage Amazon S3 Glacier, la définition de la propriété de la table `read_restored_glacier_objects` sur `false` n'a aucun effet.

  Par exemple, supposons que vous exécutiez la commande suivante :

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

  Pour les tables Iceberg et Delta Lake, la commande produit l'erreur Unsupported table property key: read_restored_glacier_objects. Pour les tables Hudi, la commande `ALTER TABLE` ne produit pas d'erreur, mais les objets Amazon S3 Glacier ne sont toujours pas ignorés. L'exécution de requêtes `SELECT` après la commande `ALTER TABLE` continue de renvoyer tous les objets.

Ressources supplémentaires

Pour des ressources supplémentaires sur l'utilisation d'Apache Hudi avec Athena, consultez les ressources suivantes.

Vidéo

La vidéo suivante montre comment utiliser Amazon Athena pour interroger un jeu de données Apache Hudi à lecture optimisée dans votre lac de données basé sur Simple Storage Service (Amazon S3).

Interrogation des jeux de données Apache Hudi à l'aide d'Amazon Athena

Billets de blogs

Les articles suivants du blog AWS Big Data décrivent comment vous pouvez utiliser Apache Hudi avec Athena.

- Utilisez AWS Data Exchange pour partager facilement les ensembles de données Apache Hudi
- Créez un lac de données near-real-time transactionnel basé sur Apache Hudi à l'aide d'Amazon AWS DMS Kinesis, du AWS Glue streaming ETL et de la visualisation des données à l'aide d'Amazon QuickSight

Création de tables Hudi

Cette section fournit des exemples d'instructions CREATE TABLE dans Athena pour les tables partitionnées et non partitionnées de données Hudi.

Si vous avez déjà créé des tables Hudi dans AWS Glue, vous pouvez les interroger directement dans Athena. Lorsque vous créez des tables Hudi partitionnées dans Athena, vous devez exécuter `ALTER TABLE ADD PARTITION` pour charger les données Hudi afin de pouvoir les interroger.

Exemples de création de table par copie sur écriture (CoW)

Table CoW non partitionnée

L'exemple suivant crée une table CoW non partitionnée dans Athena.

```
CREATE EXTERNAL TABLE `non_partition_cow`(
```

```
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int,
  `event_type` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/non_partition_cow/'
```

Table CoW partitionnée

L'exemple suivant crée une table CoW partitionnée dans Athena.

```
CREATE EXTERNAL TABLE `partition_cow`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int)
PARTITIONED BY (
  `event_type` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/partition_cow/'
```

L'exemple `ALTER TABLE ADD PARTITION` suivant ajoute deux partitions à la table `partition_cow` en exemple.

```
ALTER TABLE partition_cow ADD
  PARTITION (event_type = 'one') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_cow/one/'
  PARTITION (event_type = 'two') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_cow/two/'
```

Exemples de création de table par fusion sur lecture (MoR)

Hudi crée deux tables dans le métastore pour MoR : une table pour les requêtes d'instantané et une table pour les requêtes à lecture optimisée. Les deux tables peuvent être interrogées. Dans les versions de Hudi antérieures à 0.5.1, la table pour les requêtes à lecture optimisée avait le nom que vous aviez spécifié lorsque vous avez créé la table. À partir de la version 0.5.1 de Hudi, le nom de la table est suffixé par défaut par le suffixe `_ro`. Le nom de la table pour les requêtes d'instantané est le nom que vous avez spécifié, suivi de `_rt`.

Table non partitionnée à fusion sur lecture (MoR)

L'exemple suivant crée une table MoR non partitionnée dans Athena pour les requêtes à lecture optimisée. Notez que les requêtes à lecture optimisée utilisent le format d'entrée `HoodieParquetInputFormat`.

```
CREATE EXTERNAL TABLE `nonpartition_mor`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int,
  `event_type` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
```

```
  's3://DOC-EXAMPLE-BUCKET/folder/nonpartition_mor/'
```

L'exemple suivant crée une table MoR non partitionnée dans Athena pour les requêtes d'instantané. Pour les requêtes d'instantané, utilisez le format d'entrée `HoodieParquetRealtimeInputFormat`.

```
CREATE EXTERNAL TABLE `nonpartition_mor_rt`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int,
  `event_type` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.realtime.HoodieParquetRealtimeInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/nonpartition_mor/'
```

Table partitionnée à fusion sur lecture (MoR)

L'exemple suivant crée une table MoR partitionnée dans Athena pour les requêtes à lecture optimisée.

```
CREATE EXTERNAL TABLE `partition_mor`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int)
PARTITIONED BY (
  `event_type` string)
ROW FORMAT SERDE
```

```
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/partition_mor/'
```

L'exemple `ALTER TABLE ADD PARTITION` suivant ajoute deux partitions à la table `partition_mor` en exemple.

```
ALTER TABLE partition_mor ADD
  PARTITION (event_type = 'one') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_mor/one/'
  PARTITION (event_type = 'two') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_mor/two/'
```

L'exemple suivant crée une table MoR partitionnée dans Athena pour les requêtes d'instantané.

```
CREATE EXTERNAL TABLE `partition_mor_rt`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int)
PARTITIONED BY (
  `event_type` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.realtime.HoodieParquetRealtimeInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/partition_mor/'
```

De même, l'exemple `ALTER TABLE ADD PARTITION` suivant ajoute deux partitions à la table `partition_mor_rt` en exemple.

```
ALTER TABLE partition_mor_rt ADD
  PARTITION (event_type = 'one') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_mor/one/'
  PARTITION (event_type = 'two') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_mor/two/'
```

Ressources supplémentaires

- Pour plus d'informations sur l'utilisation de connecteurs AWS Glue personnalisés et de tâches AWS Glue 2.0 pour créer une table Apache Hudi que vous pouvez interroger avec Athena, consultez la section Écrire dans des tables Apache Hudi à AWS Glue l'aide d'un connecteur personnalisé sur AWS le blog Big Data.
- Pour un article sur l'utilisation d'Apache Hudi et d'Amazon Athena pour créer une infrastructure de traitement des données pour un lac de données, consultez Simplifier le traitement des données opérationnelles dans les lacs de données à AWS Glue l'aide d'Apache Hudi et Apache Hudi dans AWS le blog Big Data. AWS Glue

Utilisation des tables Apache Iceberg

Athena prend en charge les requêtes de lecture, de voyage dans le temps, d'écriture et DDL pour les tables Apache Iceberg qui utilisent le format Apache Parquet pour les données et le AWS Glue catalogue de leur métastore.

Apache Iceberg est un format de table ouvert pour les jeux de données analytiques très volumineux. Iceberg gère de vastes collections de fichiers sous forme de tables et prend en charge les opérations modernes de lac de données analytiques telles que les requêtes d'insertion, de mise à jour, de suppression et Time Travel au niveau des enregistrements. La spécification Iceberg permet une évolution transparente des tables, comme l'évolution des schémas et des partitions, et est conçue pour une utilisation optimisée sur Amazon S3. Iceberg contribue également à garantir l'exactitude des données dans des scénarios d'écriture simultanés.

Pour plus d'informations sur Apache Iceberg, consultez https://iceberg.apache.org/.

Considérations et restrictions

La prise en charge par Athena des tables Iceberg comporte les considérations et limites suivantes :

- Support de version Iceberg — Athena prend en charge la version 1.4.2 d'Apache Iceberg.

- Tables avec AWS Glue catalogue uniquement : seules les tables Iceberg créées à partir du AWS Glue catalogue sur la base des spécifications définies par l'implémentation du catalogue Glue open source sont prises en charge par Athena.
- Support de verrouillage des tables AWS Glue uniquement : contrairement à l'implémentation du catalogue open source Glue, qui prend en charge le verrouillage personnalisé par plug-in, Athena prend uniquement en charge le verrouillage AWS Glue optimiste. L'utilisation d'Athena pour modifier une table Iceberg avec n'importe quelle autre implémentation de verrouillage entraînera une perte de données potentielle et interrompra les transactions.
- Formats de fichier pris en charge – La prise en charge des formats de fichier Iceberg dans Athena dépend de la version du moteur Athena, comme indiqué dans le tableau suivant.

Version du moteur Athena	Parquet	ORC	Avro
2	Oui	Non	Non
3	Oui	Oui	Oui

- Tables Iceberg v2 : Athena ne crée et n'opère que sur des tables Iceberg v2. Pour connaître la différence entre les tables v1 et v2, consultez la section Modifications de version de format dans la documentation Apache Iceberg.
- Affichage des types horaires sans fuseau horaire – L'heure et l'horodatage sans types de fuseau horaire sont affichés en UTC. Si le fuseau horaire n'est pas spécifié dans une expression de filtre sur une colonne de temps, UTC est utilisé.
- Précision des données liées à l'horodatage – Bien qu'Iceberg prenne en charge la précision à la microseconde pour le type de données d'horodatage, Athena ne prend en charge que la précision à la milliseconde pour les horodatages, tant en lecture qu'en écriture. Pour les données contenues dans les colonnes liées au temps réécrites lors des opérations de compactage manuel, Athena conserve uniquement la précision a la milliseconde.
- Opérations non prises en charge : les opérations Athena suivantes ne sont pas prises en charge pour les tables Iceberg.
  - ALTER TABLE SET LOCATION
- Vues – Utiliser `CREATE VIEW` pour créer des vues Athena comme décrit dans Utilisation des vues. Si vous souhaitez utiliser la spécification de vue Iceberg pour créer des vues, contactez athena-feedback@amazon.com.

- Les commandes de gestion TTF ne sont pas prises en charge dans AWS Lake Formation — Bien que vous puissiez utiliser Lake Formation pour gérer les autorisations d'accès en lecture pour des TransactionTable formats (TTF) tels qu'Apache Iceberg, Apache Hudi et Linux Foundation Delta Lake, vous ne pouvez pas utiliser Lake Formation pour gérer les autorisations pour des opérations telles que `UPDATE` ou `OPTIMIZE` avec `VACUUM` ces `MERGE` formats de table. Pour plus d'informations sur l'intégration de Lake Formation à Athena, consultez la section Utilisation AWS Lake Formation avec Amazon Athena dans AWS Lake Formation le manuel du développeur.
- Partitionnement par champs imbriqués – Le partitionnement par champs imbriqués n'est pas pris en charge. Toute tentative de ce type génère le message NOT_SUPPORTED : Le partitionnement par champ imbriqué n'est pas pris en charge : *`column_name.nested_field_name`*.
- Omission d'objets S3 Glacier non prise en charge : si les objets de la table Apache Iceberg appartiennent à une classe de stockage Amazon S3 Glacier, la définition de la propriété de la table `read_restored_glacier_objects` sur `false` n'a aucun effet.

  Par exemple, supposons que vous exécutiez la commande suivante :

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

  Pour les tables Iceberg et Delta Lake, la commande produit l'erreur Unsupported table property key: read_restored_glacier_objects. Pour les tables Hudi, la commande `ALTER TABLE` ne produit pas d'erreur, mais les objets Amazon S3 Glacier ne sont toujours pas ignorés. L'exécution de requêtes `SELECT` après la commande `ALTER TABLE` continue de renvoyer tous les objets.

Si vous souhaitez qu'Athena prenne en charge une fonction particulière, envoyez vos commentaires à l'adresse athena-feedback@amazon.com.

Rubriques

- Autres opérations Athena sur les tables Iceberg
- Ressources supplémentaires

Création de tables Iceberg

Pour créer une table Iceberg à utiliser dans Athena, vous pouvez utiliser `CREATE TABLE` une instruction telle que décrite sur cette page, ou vous pouvez utiliser AWS Glue un robot d'exploration.

Utilisation d'une instruction CREATE TABLE

Athena crée des tables Iceberg v2. Pour connaître la différence entre les tables v1 et v2, consultez la section Modifications de version de format dans la documentation Apache Iceberg.

Athena `CREATE TABLE` crée une table Iceberg sans données. Vous pouvez interroger une table à partir de systèmes externes tels qu'Apache Spark directement si la table utilise le catalogue open source Glue d'Iceberg. Il n'est pas nécessaire de créer une table externe.

> **Warning**
>
> L'exécution de `CREATE EXTERNAL TABLE` entraîne le message d'erreur External keyword not supported for table type ICEBERG (Mot clé externe non pris en charge pour le type de table ICEBERG).

Pour créer une table Iceberg à partir d'Athena, définissez la propriété de table `'table_type'` à `'ICEBERG'` dans la clause `TBLPROPERTIES`, comme dans le récapitulatif de la syntaxe suivant.

```
CREATE TABLE
  [db_name.]table_name (col_name data_type [COMMENT col_comment] [, ...] )
  [PARTITIONED BY (col_name | transform, ... )]
  LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
  TBLPROPERTIES ( 'table_type' ='ICEBERG' [, property_name=property_value] )
```

Pour plus d'informations sur les types de données que vous pouvez interroger dans les tables Iceberg, consultez Types de données pris en charge pour les tables Iceberg sur Athena.

Partitioning

Pour créer des tables Iceberg avec des partitions, utilisez la syntaxe `PARTITIONED BY`. Les colonnes utilisées pour le partitionnement doivent d'abord être spécifiées dans les déclarations de

colonnes. Dans la clause `PARTITIONED BY`, le type de colonne ne doit pas être inclus. Vous pouvez également définir des transformations de partition dans la syntaxe `CREATE TABLE`. Pour spécifier plusieurs colonnes à partitionner, séparez les colonnes par la virgule (`,`), comme dans l'exemple suivant.

```
CREATE TABLE iceberg_table (id bigint, data string, category string)
  PARTITIONED BY (category, bucket(16, id))
  LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
  TBLPROPERTIES ( 'table_type' = 'ICEBERG' )
```

Le tableau suivant présente les fonctions de transformation de partition disponibles.

Fonction	Description	Types pris en charge
`year(ts)`	Partition par année	`date`, `timestamp`
`month(ts)`	Partition par mois	`date`, `timestamp`
`day(ts)`	Partition par jour	`date`, `timestamp`
`hour(ts)`	Partition par heure	`timestamp`
`bucket(N, col)`	Partition par compartiments de valeur hachée mod *N*. C'est le même concept que le compartiment de hachage pour les tables Hive.	`int`, `long`, `decimal`, `date`, `timestamp` , `string`, `binary`
`truncate( L, col)`	Partition par valeur tronquée à *L*	`int`, `long`, `decimal`, `string`

Athena prend en charge le partitionnement caché d'Iceberg. Pour plus d’informations, consultez Partitionnement caché d'Iceberg dans la documentation Apache Iceberg.

Propriétés de la table

Cette section décrit les propriétés de la table que vous pouvez spécifier sous forme de paires clé-valeur dans la clause `TBLPROPERTIES` de l'instruction `CREATE TABLE`. Athena n'autorise qu'une liste prédéfinie de paires clé-valeur dans les propriétés de la table pour créer ou modifier des tables Iceberg. Les tableaux suivants présentent les propriétés de la table que vous pouvez spécifier. Pour plus d'informations sur les options de compactage, consultez la section Optimisation des tables Iceberg dans cette documentation. Si vous souhaitez qu'Athena prenne en charge une propriété de configuration de table open source spécifique, envoyez vos commentaires à athena-feedback@amazon.com.

format

Description	Format de données du fichier
Valeurs de propriété autorisées	Le format de fichier et les combinaisons de compression pris en charge varient selon la version du moteur Athena. Pour plus d'informations, consultez Prise en charge de la compression de la table Iceberg par format de fichier.
Valeur par défaut	parquet

write_compression

Description	Codec de compression de fichier
Valeurs de propriété autorisées	Le format de fichier et les combinaisons de compression pris en charge varient selon la version du moteur Athena. Pour plus d'informations, consultez Prise en charge de la compression de la table Iceberg par format de fichier.
Valeur par défaut	La compression d'écriture par défaut varie selon la version du moteur Athena. Pour plus d'informations, consultez Prise en charge de la compression de la table Iceberg par format de fichier.

optimize_rewrite_data_file_threshold

Description	Configuration spécifique à l'optimisation des données. S'il y a moins de fichiers de données nécessitant une optimisation que le seuil donné, les fichiers ne sont pas réécrits. Cela permet d'accumuler davantage de fichiers de données pour produire des fichiers plus proches de la taille cible et ignorer les calculs inutiles pour économiser les coûts.
Valeurs de propriété autorisées	Nombre positif. Il doit être inférieur à 50.
Valeur par défaut	5

optimize_rewrite_delete_file_seuil

Description	Configuration spécifique à l'optimisation des données. S'il y a moins de fichiers de suppression associés à un fichier de données que le seuil, le fichier de données n'est pas réécrit. Cela permet d'accumuler davantage de fichiers supprimés pour chaque fichier de données afin de réaliser des économies.
Valeurs de propriété autorisées	Nombre positif. Il doit être inférieur à 50.
Valeur par défaut	2

vacuum_min_snapshots_to_keep

Description	Nombre minimum d'instantanés à retenir sur la branche principale d'une table. Cette valeur est prioritaire sur la propriété `vacuum_max_snapshot_age_seconds` . Si le nombre minimal d'instantanés restants est supérieur à l'âge spécifié par `vacuum_max_snapshot_age_seconds` , les instantanés sont conservés et la valeur de `vacuum_max_snapshot_age_seconds` est ignorée.

Valeurs de propriété autorisées	Nombre positif.
Valeur par défaut	1

vacuum_max_snapshot_age_seconds

Description	Âge maximum des instantanés à retenir sur la branche principale. Cette valeur est ignorée si le minimum d'instantanés restant spécifié par `vacuum_min_snapshots_to_keep` est supérieur à l'âge spécifié. Cette propriété de comportement de table correspond à la `history.expire.max-snapshot-age-ms` propriété de la configuration d'Apache Iceberg.
Valeurs de propriété autorisées	Nombre positif.
Valeur par défaut	432 000 secondes (5 jours)

vacuum_max_metadata_files_to_keep

Description	Nombre maximal de fichiers de métadonnées précédents à conserver dans la branche principale de la table.
Valeurs de propriété autorisées	Nombre positif.
Valeur par défaut	100

Exemple d'instruction CREATE TABLE (CRÉER UNE TABLE)

L'exemple suivant crée une table Iceberg à trois colonnes.

```
CREATE TABLE iceberg_table (
  id int,
  data string,
```

```
  category string)
PARTITIONED BY (category, bucket(16,id))
LOCATION 's3://DOC-EXAMPLE-BUCKET/iceberg-folder'
TBLPROPERTIES (
  'table_type'='ICEBERG',
  'format'='parquet',
  'write_compression'='snappy',
  'optimize_rewrite_delete_file_threshold'='10'
)
```

CREATE TABLE AS SELECT (CTAS)

Pour plus d'informations sur la création d'une table Iceberg à l'aide de l'instruction `CREATE TABLE AS`, veuillez consulter CREATE TABLE AS, en portant une attention particulière à la section Propriétés de la table CTAS.

Utilisation d'un Crawler AWS Glue

Vous pouvez utiliser un AWS Glue robot pour enregistrer automatiquement vos tables Iceberg dans le. AWS Glue Data Catalog Si vous souhaitez effectuer une migration depuis un autre catalogue Iceberg, vous pouvez créer et planifier un AWS Glue robot et fournir les chemins Amazon S3 où se trouvent les tables Iceberg. Vous pouvez spécifier la profondeur maximale des chemins Amazon S3 que le Crawler AWS Glue peut parcourir. Une fois que vous avez planifié un AWS Glue robot d'exploration, celui-ci extrait les informations du schéma et les met à jour en fonction des modifications apportées AWS Glue Data Catalog au schéma à chaque fois qu'il s'exécute. Le AWS Glue robot d'exploration prend en charge la fusion de schémas entre les instantanés et met à jour l'emplacement du dernier fichier de métadonnées dans le. AWS Glue Data Catalog Pour plus d'informations, consultez la section Catalogue de données et robots d'exploration dans AWS Glue.

Gestion des tables Iceberg

Athena prend en charge les opérations DDL de table suivantes pour les tables Iceberg.

ALTER TABLE RENAME

Renomme une table.

Étant donné que les métadonnées de table d'une table Iceberg sont stockées dans Simple Storage Service (Amazon S3), vous pouvez mettre à jour la base de données et le nom de la table d'une table gérée Iceberg sans affecter les informations de table sous-jacentes.

Résumé

```
ALTER TABLE [db_name.]table_name RENAME TO [new_db_name.]new_table_name
```

Exemple

```
ALTER TABLE my_db.my_table RENAME TO my_db2.my_table2
```

ALTER TABLE SET PROPERTIES

Ajoute des propriétés à une table Iceberg et définit les valeurs qui leur sont attribuées.

Conformément aux spécifications Iceberg, les propriétés de la table sont stockées dans le fichier de métadonnées de la table Iceberg plutôt que dans AWS Glue. Athena n'accepte pas les propriétés de table personnalisées. Reportez-vous à la section Propriétés de la table en ce qui concerne les paires clé-valeur autorisées. Si vous souhaitez qu'Athena prenne en charge une propriété de configuration de table open source spécifique, envoyez vos commentaires à athena-feedback@amazon.com.

Résumé

```
ALTER TABLE [db_name.]table_name SET TBLPROPERTIES ('property_name' =
 'property_value' [ , ... ])
```

Exemple

```
ALTER TABLE iceberg_table SET TBLPROPERTIES (
  'format'='parquet',
  'write_compression'='snappy',
  'optimize_rewrite_delete_file_threshold'='10'
)
```

ALTER TABLE UNSET PROPERTIES

Supprime les propriétés existantes d'une table Iceberg.

Résumé

```
ALTER TABLE [db_name.]table_name UNSET TBLPROPERTIES ('property_name' [ , ... ])
```

Exemple

```
ALTER TABLE iceberg_table UNSET TBLPROPERTIES ('write_compression')
```

DESCRIBE TABLE

Décrit les informations du tableau.

Résumé

```
DESCRIBE [FORMATTED] [db_name.]table_name
```

Lorsque l'option `FORMATTED` est spécifiée, la sortie affiche des informations supplémentaires telles que l'emplacement de la table et les propriétés.

Exemple

```
DESCRIBE iceberg_table
```

DROP TABLE

Supprime une table Iceberg.

> **Warning**
>
> Étant donné que les tables Iceberg sont considérées comme des tables gérées dans Athena, la suppression d'une table Iceberg supprime également toutes les données de la table.

Résumé

```
DROP TABLE [IF EXISTS] [db_name.]table_name
```

Exemple

```
DROP TABLE iceberg_table
```

SHOW CREATE TABLE

Affiche une instruction DDL `CREATE TABLE` qui peut être utilisée pour recréer la table Iceberg dans Athena. Si Athena ne peut pas reproduire la structure de la table (par exemple, parce que les

propriétés de table personnalisées sont spécifiées dans la table), une erreur UNSUPPORTED (NON PRIS EN CHARGE) est générée.

Résumé

```
SHOW CREATE TABLE [db_name.]table_name
```

Exemple

```
SHOW CREATE TABLE iceberg_table
```

SHOW TABLE PROPERTIES

Affiche une ou plusieurs propriétés de table d'une table Iceberg. Seules les propriétés de table prises en charge par Athena s'affichent.

Résumé

```
SHOW TBLPROPERTIES [db_name.]table_name [('property_name')]
```

Exemple

```
SHOW TBLPROPERTIES iceberg_table
```

Interrogation des métadonnées d'une table Iceberg

Dans une requête SELECT, vous pouvez utiliser les propriétés suivantes après *table_name* pour interroger les métadonnées de la table Iceberg :

- $files – Affiche les fichiers de données actuels d'une table.
- $manifest – Affiche les manifestes des fichiers actuels d'une table.
- $history – Affiche l'historique d'une table.
- $partitions – Affiche les partitions actuelles d'une table.
- $snapshots – Affiche les instantanés d'une table.
- $refs – Affiche les références d'une table.

Syntaxe

L'instruction suivante répertorie les fichiers d'une table Iceberg.

```
SELECT * FROM "dbname"."tablename$files"
```

L'instruction suivante répertorie les manifestes d'une table Iceberg.

```
SELECT * FROM "dbname"."tablename$manifests"
```

L'instruction suivante affiche l'historique d'une table Iceberg.

```
SELECT * FROM "dbname"."tablename$history"
```

L'exemple suivant affiche les partitions d'une table Iceberg.

```
SELECT * FROM "dbname"."tablename$partitions"
```

L'exemple suivant affiche les instantanés d'une table Iceberg.

```
SELECT * FROM "dbname"."tablename$snapshots"
```

L'exemple suivant affiche les références d'une table Iceberg.

```
SELECT * FROM "dbname"."tablename$refs"
```

Schéma évolutif de la table Iceberg

Les mises à jour du schéma Iceberg sont des modifications de métadonnées uniquement. Aucun fichier de données n'est modifié lorsque vous effectuez une mise à jour du schéma.

Le format Iceberg prend en charge les modifications suivantes de l'évolution du schéma :

- Add (Ajouter) – Ajoute une nouvelle colonne à une table ou à un `struct` imbriqué.
- Drop (Supprimer) – Supprime une colonne existante d'une table ou d'un `struct` imbriqué.
- Rename (Renommer) – Renomme une colonne ou un champ existant dans un `struct` imbriqué.
- Reorder (Réorganiser) – Modifie l'ordre des colonnes.
- Type promotion (Promotion de type) – Élargit le type d'une colonne, d'un champ `struct`, d'une clé `map`, d'une valeur `map`, ou d'un élément `list`. Actuellement, les cas suivants sont pris en charge pour les tables Iceberg :
  - entier à grand nombre entier

- float à double
- augmentation de la précision d'un type décimal

ALTER TABLE ADD COLUMNS

Ajoute une ou plusieurs colonnes à une table Iceberg existante.

Résumé

```
ALTER TABLE [db_name.]table_name ADD COLUMNS (col_name data_type [,...])
```

Exemples

L'exemple suivant ajoute une colonne `comment` de type `string` sur une table Iceberg.

```
ALTER TABLE iceberg_table ADD COLUMNS (comment string)
```

L'exemple suivant ajoute une colonne `point` de type `struct` sur une table Iceberg.

```
ALTER TABLE iceberg_table
ADD COLUMNS (point struct<x: double, y: double>)
```

L'exemple suivant ajoute une colonne `points` qui est un tableau de structures à une table Iceberg.

```
ALTER TABLE iceberg_table
ADD COLUMNS (points array<struct<x: double, y: double>>)
```

ALTER TABLE DROP COLUMN

Supprime une colonne d'une table Iceberg existante.

Résumé

```
ALTER TABLE [db_name.]table_name DROP COLUMN col_name
```

Exemple

```
ALTER TABLE iceberg_table DROP COLUMN userid
```

ALTER TABLE CHANGE COLUMN

Modifie le nom, le type, l'ordre ou le commentaire d'une colonne.

> **Note**
>
> `ALTER TABLE REPLACE COLUMNS` n'est pas pris en charge. Étant donné que `REPLACE COLUMNS` supprime toutes les colonnes, puis en ajoute de nouvelles, elle n'est pas prise en charge pour Iceberg. `CHANGE COLUMN` est la syntaxe préférée pour l'évolution du schéma.

Résumé

```
ALTER TABLE [db_name.]table_name
  CHANGE [COLUMN] col_old_name col_new_name column_type
  [COMMENT col_comment] [FIRST|AFTER column_name]
```

Exemple

```
ALTER TABLE iceberg_table CHANGE comment blog_comment string AFTER id
```

SHOW COLUMNS

Affiche les colonnes d'un tableau.

Résumé

```
SHOW COLUMNS (FROM|IN) [db_name.]table_name
```

Exemple

```
SHOW COLUMNS FROM iceberg_table
```

Interrogation des données de la table et exécution de Time Travel

Pour interroger un jeu de données Iceberg, utilisez une instruction standard `SELECT` comme suit. Les requêtes suivent les spécifications du format Apache Iceberg v2 et effectuent merge-on-read des suppressions de position et d'égalité.

```
SELECT * FROM [db_name.]table_name [WHERE predicate]
```

Pour optimiser les temps de requête, certains prédicats sont poussés vers l'endroit où résident les données.

Requêtes Time Travel et Version Travel

Chaque table Apache Iceberg conserve un manifeste versionné des objets Simple Storage Service (Amazon S3) qu'elle contient. Les versions précédentes du manifeste peuvent être utilisées pour les requêtes Time Travel et Version Travel.

Les requêtes Time Travel dans Athena interrogent Simple Storage Service (Amazon S3) des données historiques à partir d'un instantané cohérent à partir d'une date et d'une heure spécifiées. Les requêtes Version Travel dans Athena interrogent Simple Storage Service (Amazon S3) pour des données historiques à partir d'un ID d'instantané spécifié.

Requêtes Time Travel

Pour exécuter une requête Time Travel, utilisez `FOR TIMESTAMP AS OF timestamp` après le nom de la table dans l'instruction `SELECT`, comme dans l'exemple suivant.

```
SELECT * FROM iceberg_table FOR TIMESTAMP AS OF timestamp
```

L'heure système à spécifier pour les déplacements est soit un horodatage, soit un horodatage avec un fuseau horaire. Si elle n'est pas spécifiée, Athena considère que la valeur est un horodatage en heure UTC.

Les exemples de requêtes de voyage dans le temps suivants sélectionnent CloudTrail des données pour la date et l'heure spécifiées.

```
SELECT * FROM iceberg_table FOR TIMESTAMP AS OF TIMESTAMP '2020-01-01 10:00:00 UTC'
```

```
SELECT * FROM iceberg_table FOR TIMESTAMP AS OF (current_timestamp - interval '1' day)
```

Requêtes Version Travel

Pour exécuter une requête Version Travel (c'est-à-dire afficher un instantané cohérent à partir d'une version spécifiée), utilisez `FOR VERSION AS OF version` après le nom de la table dans l'instruction `SELECT`, comme dans l'exemple suivant.

```
SELECT * FROM [db_name.]table_name FOR VERSION AS OF version
```

Le paramètre *version* est l'ID de l'instantané `bigint` associé à une version de table Iceberg.

L'exemple de requête Version Travel suivant sélectionne les données pour la version spécifiée.

```
SELECT * FROM iceberg_table FOR VERSION AS OF 949530903748831860
```

> **Note**
>
> Les clauses `FOR SYSTEM_TIME AS OF` et `FOR SYSTEM_VERSION AS OF` de la version 2 du moteur Athena ont été remplacées par les clauses `FOR TIMESTAMP AS OF` et `FOR VERSION AS OF` de la version 3 du moteur Athena.

Récupération de l'ID d'instantané

Vous pouvez utiliser la SnapshotUtilclasse Java fournie par Iceberg pour récupérer l'ID du snapshot Iceberg, comme dans l'exemple suivant.

```
import org.apache.iceberg.Table;
import org.apache.iceberg.aws.glue.GlueCatalog;
import org.apache.iceberg.catalog.TableIdentifier;
import org.apache.iceberg.util.SnapshotUtil;

import java.text.SimpleDateFormat;
import java.util.Date;

Catalog catalog = new GlueCatalog();

Map<String, String> properties = new HashMap<String, String>();
properties.put("warehouse", "s3://DOC-EXAMPLE-BUCKET/my-folder");
catalog.initialize("my_catalog", properties);

Date date = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").parse("2022/01/01 00:00:00");
long millis = date.getTime();

TableIdentifier name = TableIdentifier.of("db", "table");
Table table = catalog.loadTable(name);
long oldestSnapshotIdAfter2022 = SnapshotUtil.oldestAncestorAfter(table, millis);
```

Combinaison de Time Travel et Version Travel

Vous pouvez utiliser la syntaxe Time Travel et Version Travel dans la même requête pour spécifier différentes conditions de synchronisation et de gestion des versions, comme dans l'exemple suivant.

```
SELECT table1.*, table2.* FROM
  [db_name.]table_name FOR TIMESTAMP AS OF (current_timestamp - interval '1' day) AS
 table1
  FULL JOIN
  [db_name.]table_name FOR VERSION AS OF 5487432386996890161 AS table2
  ON table1.ts = table2.ts
  WHERE (table1.id IS NULL OR table2.id IS NULL)
```

Création et interrogation de vues à l'aide de tables Iceberg

Pour créer et interroger des vues Athena sur des tables Iceberg, utilisez des vues `CREATE VIEW` comme décrit dans Utilisation des vues.

Exemple :

```
CREATE VIEW view1 AS SELECT * FROM iceberg_table
```

```
SELECT * FROM view1
```

Si vous souhaitez utiliser la spécification de vue Iceberg pour créer des vues, contactez athena-feedback@amazon.com.

Utilisation du contrôle d'accès précis de Lake Formation

La version 3 du moteur Athena prend en charge le contrôle d'accès précis de Lake Formation avec les tables Iceberg, y compris le contrôle d'accès de sécurité au niveau des colonnes et des lignes. Ce contrôle d'accès fonctionne avec les requêtes de voyage dans le temps et avec les tables qui ont effectué une évolution de leur schéma. Pour plus d'informations, consultez Contrôle d'accès précis de Lake Formation et groupes de travail Athena.

Si vous avez créé votre table Iceberg en dehors d'Athena, utilisez le kit SDK Apache Iceberg en version 0.13.0 ou supérieure pour que les informations des colonnes de votre table Iceberg soient renseignées dans le AWS Glue Data Catalog. Si votre table Iceberg ne contient pas d'informations de colonne AWS Glue, vous pouvez utiliser l'instruction ALTER TABLE SET PROPERTIES Athena ou

le dernier SDK Iceberg pour corriger la table et mettre à jour les informations de colonne dans. AWS Glue

Mise à jour des données de la table Iceberg

Les données des tables Iceberg peuvent être gérées directement sur Athena à l'aide des requêtes `INSERT`, `UPDATE` et `DELETE`. Chaque transaction de gestion des données produit un nouvel instantané, qui peut être interrogé à l'aide de requêtes Time Travel. Les instructions `UPDATE` et `DELETE` suivent la spécification de suppression de position au niveau des lignes du format Iceberg v2 et appliquent l'isolation des instantanés.

Utilisez les commandes suivantes pour effectuer des opérations de gestion des données sur les tables Iceberg.

INSERT INTO

Insère des données dans une table Iceberg. Athena Iceberg `INSERT INTO` est facturé de la même manière que les requêtes `INSERT INTO` actuelles pour les tables Hive externes, en fonction de la quantité de données analysées. Pour insérer des données dans une table Iceberg, utilisez la syntaxe suivante, où la *`query`* (requête) peut être soit `VALUES (val1, val2, ...)` ou `SELECT (col1, col2, …) FROM [`*`db_name`*`.]`*`table_name`* `WHERE` *`predicate`*. Pour des détails sur la syntaxe SQL et la sémantique, voir INSERT INTO.

```
INSERT INTO [db_name.]table_name [(col1, col2, …)] query
```

Les exemples suivants insèrent des valeurs dans la table `iceberg_table`.

```
INSERT INTO iceberg_table VALUES (1,'a','c1')
```

```
INSERT INTO iceberg_table (col1, col2, ...) VALUES (val1, val2, ...)
```

```
INSERT INTO iceberg_table SELECT * FROM another_table
```

DELETE

Athena Iceberg `DELETE` écrit les fichiers de suppression de position Iceberg dans une table. C'est ce que l'on appelle une merge-on-readsuppression. Contrairement à une copy-on-writesuppression, celle-ci est plus efficace car elle ne réécrit pas les données du fichier. merge-on-read Lorsque Athena

lit les données Iceberg, elle fusionne les fichiers de suppression de position Iceberg avec des fichiers de données pour produire la dernière vue d'un tableau. Pour supprimer ces fichiers de suppression de position, vous pouvez exécuter l'action de compactage REWRITE DATA (RÉÉCRITURE DES DONNÉES). Les opérations `DELETE` sont facturées en fonction de la quantité de données analysées. Pour la syntaxe, consultez DELETE.

L'exemple suivant supprime les lignes de `iceberg_table` qui ont `c3` comme valeur pour `category`.

```
DELETE FROM iceberg_table WHERE category='c3'
```

UPDATE

Athena Iceberg `UPDATE` écrit les fichiers de suppression de position Iceberg et les nouvelles lignes mises à jour en tant que fichiers de données dans la même transaction.`UPDATE` peut être imaginé comme une combinaison de `INSERT INTO` et `DELETE`. Les opérations `UPDATE` sont facturées en fonction de la quantité de données analysées. Pour la syntaxe, consultez MISE A JOUR.

L'exemple suivant met à jour les valeurs spécifiées dans la table `iceberg_table`.

```
UPDATE iceberg_table SET category='c2' WHERE category='c1'
```

MERGE INTO

Mise à jour, suppression ou insertion conditionnelle de lignes dans une table Iceberg. Une seule instruction peut combiner des actions de mise à jour, de suppression et d'insertion. Pour la syntaxe, consultez MERGE INTO.

> **Note**
>
> `MERGE INTO` est transactionnel et n'est pris en charge que pour les tables Apache Iceberg dans la version 3 du moteur Athena.

L'exemple suivant supprime tous les clients de la table `t` qui se trouvent dans la table source `s`.

```
MERGE INTO accounts t USING monthly_accounts_update s
ON t.customer = s.customer
```

```
WHEN MATCHED
THEN DELETE
```

L'exemple suivant met à jour la table cible `t` avec les informations des clients provenant de la table source `s`. Pour les lignes de clients dans la table `t` qui ont des lignes de clients correspondantes dans la table `s`, l'exemple incrémente les achats dans la table t. Si la table `t` ne correspond pas à une ligne de client dans la table `s`, l'exemple insère la ligne de client de la table `s` dans la table `t`.

```
MERGE INTO accounts t USING monthly_accounts_update s
    ON (t.customer = s.customer)
    WHEN MATCHED
        THEN UPDATE SET purchases = s.purchases + t.purchases
    WHEN NOT MATCHED
        THEN INSERT (customer, purchases, address)
              VALUES(s.customer, s.purchases, s.address)
```

L'exemple suivant met à jour de manière conditionnelle la table cible `t` avec les informations de la table source `s`. L'exemple supprime toute ligne cible correspondante dont l'adresse source est Centreville. Pour toutes les autres lignes correspondantes, l'exemple ajoute les achats source et définit l'adresse cible comme adresse source. S'il n'y a aucune correspondance dans la table cible, l'exemple insère la ligne de la table source.

```
MERGE INTO accounts t USING monthly_accounts_update s
    ON (t.customer = s.customer)
    WHEN MATCHED AND s.address = 'Centreville'
        THEN DELETE
    WHEN MATCHED
        THEN UPDATE
            SET purchases = s.purchases + t.purchases, address = s.address
    WHEN NOT MATCHED
        THEN INSERT (customer, purchases, address)
              VALUES(s.customer, s.purchases, s.address)
```

Optimisation des tables Iceberg

À mesure que les données s'accumulent dans une table Iceberg, les requêtes deviennent progressivement moins efficaces en raison du temps de traitement accru requis pour ouvrir les fichiers. Des coûts de calcul supplémentaires sont encourus si la table contient des fichiers de suppression. Dans Iceberg, les fichiers de suppression stockent les suppressions au niveau des lignes, et le moteur doit appliquer les lignes supprimées aux résultats de la requête.

Afin d'optimiser les performances des requêtes sur les tables Iceberg, Athena prend en charge le compactage manuel en tant que commande de maintenance de table. Les compactions optimisent la disposition structurelle de la table sans en modifier le contenu.

OPTIMIZE

L'action de compactage `OPTIMIZE table REWRITE DATA` réécrit les fichiers de données dans une disposition plus optimisée en fonction de leur taille et du nombre de fichiers de suppression associés. Pour plus de détails sur la syntaxe et les propriétés des tables, voir OPTIMIZE.

Exemple

L'exemple suivant fusionne les fichiers de suppression dans des fichiers de données et produit des fichiers proches de la taille de fichier ciblée où la valeur de `category` est `c1`.

```
OPTIMIZE iceberg_table REWRITE DATA USING BIN_PACK
  WHERE category = 'c1'
```

VACUUM

`VACUUM` exécute l'expiration d'instantané et la suppression des fichiers orphelins. Ces actions réduisent la taille des métadonnées et suppriment les fichiers qui ne figurent pas dans l'état actuel de la table et qui sont également plus anciens que la période de rétention spécifiée pour la table. Pour plus de détails sur la syntaxe, voir VACUUM.

Exemple

L'exemple suivant utilise une propriété de table pour configurer la table `iceberg_table` afin de retenir les données des trois derniers jours, puis utilise `VACUUM` pour faire expirer les anciens instantanés et supprimer les fichiers orphelins de la table.

```
ALTER TABLE iceberg_table SET TBLPROPERTIES (
  'vacuum_max_snapshot_age_seconds'='259200'
)

VACUUM iceberg_table
```

Types de données pris en charge pour les tables Iceberg sur Athena

Athena peut interroger des tables Iceberg contenant les types de données suivants :

```
binary
boolean
date
decimal
double
float
int
list
long
map
string
struct
timestamp without time zone
```

Pour plus d'informations sur les types de tables Iceberg, consultez la page de schémas pour Iceberg dans la documentation Apache.

Le tableau suivant illustre la relation entre les types de données Athena et les types de données de table Iceberg.

Type Iceberg	Type Athena	Remarques
`boolean`	`boolean`	
-	`tinyint`	Non pris en charge pour les tables Iceberg sur Athena.
-	`smallint`	Non pris en charge pour les tables Iceberg sur Athena.
`int`	`int`	Dans les instructions Athena DML, ce type est `INTEGER`.
`long`	`bigint`	
`double`	`double`	
`float`	`float`	
`decimal(P, S)`	`decimal(P, S)`	`P` est la précision, `S` est l'échelle.
-	`char`	Non pris en charge pour les tables Iceberg sur Athena.

Type Iceberg	Type Athena	Remarques
`string`	`string`	Dans les instructions Athena DML, ce type est `VARCHAR`.
`binary`	`binary`	
`date`	`date`	
`time`	-	Seul l'horodatage Iceberg (sans fuseau horaire) est pris en charge pour les instructions DDL Athena Iceberg telles que `CREATE TABLE`, mais tous les types d'horodatage peuvent être interrogés via Athena.
`timestamp`	`timestamp`	
`timestamp tz`	`timestamp tz`	
`list<E>`	`array`	
`map<K,V>`	`map`	
`struct<...>`	`struct`	
`fixed(L)`	-	Le type `fixed(L)` n'est actuellement pas pris en charge sur Athena.

Pour de plus amples informations sur les types de données sur Athena, consultez Types de données dans Amazon Athena.

Autres opérations Athena sur les tables Iceberg

Opérations de base de données

Lorsque vous utilisez DROP DATABASE avec l'option `CASCADE`, toutes les données de table Iceberg sont également supprimées. Les opérations DDL suivantes n'ont aucun effet sur les tables Iceberg.

- CREATE DATABASE
- ALTER DATABASE SET DBPROPERTIES
- SHOW DATABASES

- SHOW TABLES
- SHOW VIEWS

Opérations liées à la partition

Parce que les tables Iceberg utilisent un partitionnement masqué, vous n'avez pas à travailler directement avec des partitions physiques. Par conséquent, les tables Iceberg d'Athena ne prennent pas en charge les opérations DDL suivantes liées à la partition :

- SHOW PARTITIONS
- ALTER TABLE ADD PARTITION
- ALTER TABLE DROP PARTITION
- ALTER TABLE RENAME PARTITION

Si vous souhaitez voir l'évolution des partitions Iceberg dans Athena, envoyez vos commentaires à athena-feedback@amazon.com.

Déchargement des tables Iceberg

Les tables Iceberg peuvent être déchargées dans des fichiers d'un dossier sur Simple Storage Service (Amazon S3). Pour plus d’informations, veuillez consulter UNLOAD.

MSCK REPAIR

Étant donné que les tables Iceberg conservent les informations relatives à la mise en page des tables, l'exécution de MSCK REPAIR TABLE comme on le fait avec les tables Hive n'est pas nécessaire et n'est pas prise en charge.

Ressources supplémentaires

L'article suivant se trouve dans la documentation des directives AWS prescriptives.

- Utilisation de tables Apache Iceberg à l'aide d'Amazon Athena SQL

Pour des articles détaillés sur l'utilisation d'Athena avec les tables Apache Iceberg, consultez les billets suivants sur le blog AWS Big Data.

- Implémenter un processus CDC sans serveur avec Apache Iceberg à l'aide d'Amazon DynamoDB et d'Amazon Athena

- Accélérer l'ingénierie des fonctionnalités de la science des données sur les lacs de données transactionnels en utilisant Amazon Athena avec Apache Iceberg
- Créez un lac de données Apache Iceberg à l'aide d'Amazon Athena, Amazon EMR et AWS Glue
- Exécuter des mises à jour/insertions dans un lac de données en utilisant Amazon Athena et Apache Iceberg
- Créez un lac de données transactionnel à l'aide d'Apache Iceberg et partagez AWS Glue des données entre comptes à l'aide d'Amazon Athena AWS Lake Formation
- Utiliser Apache Iceberg dans un lac de données pour prendre en charge le traitement incrémentiel des données
- Créer en temps réel un lac de données Apache Iceberg aligné sur le RGPD
- Automatisez la réplication des sources relationnelles dans un lac de données transactionnel avec Apache Iceberg et AWS Glue
- Interagissez avec les tables Apache Iceberg à l'aide d'Amazon Athena et avec des autorisations détaillées entre comptes à l'aide de AWS Lake Formation
- Créer un lac de données transactionnel sans serveur avec Apache Iceberg, Amazon EMR sans serveur et Amazon Athena

Sécurité Amazon Athena

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est responsable de la protection de l'infrastructure qui fonctionne Services AWS dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. L'efficacité de notre sécurité est régulièrement testée et vérifiée par des auditeurs tiers dans le cadre des programmes de conformitéAWS. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Athena, consultez la rubrique services AWS concernés par le programme de conformité.
- Sécurité dans le cloud — Votre responsabilité est déterminée par Service AWS ce que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre organisation, et la législation et la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lorsque vous utilisez Amazon Athena. Les rubriques suivantes vous montrent comment configurer Athena pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à en utiliser d'autres Services AWS qui peuvent vous aider à surveiller et à sécuriser vos ressources Athena.

Rubriques

Protection des données dans Athena

Le modèle de responsabilité AWS partagée s'applique à la protection des données dans Amazon Athena. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.

- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-2 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS (Federal Information Processing Standard) disponibles, consultez Federal Information Processing Standard (FIPS) 140-2 (Normes de traitement de l'information fédérale).

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Name (Nom). Cela inclut lorsque vous travaillez avec Athena ou une autre personne à Services AWS l'aide de la console, de l'API ou AWS des AWS CLI SDK. Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Une étape de sécurité supplémentaire consiste à utiliser la clé contextuelle de condition globale aws:CalledVia pour limiter les requêtes à celles effectuées à partir d'Athena. Pour plus d'informations, consultez Utilisation d'Athéna avec CalledVia des touches contextuelles.

Protection de plusieurs types de données

Plusieurs types de données sont impliqués lorsque vous utilisez Athena pour créer des bases de données et des tables. Ces types de données incluent les données sources stockées dans Amazon S3, les métadonnées pour les bases de données et les tables que vous créez lorsque vous exécutez des requêtes ou le AWS Glue Crawler pour découvrir des données, les données des résultats des requêtes et l'historique des requêtes. Cette section décrit chaque type de données et fournit des conseils sur sa protection.

- Données source : vous stockez les données des bases de données et des tables dans Simple Storage Service (Amazon S3) et Athena ne les modifie pas. Pour de plus amples informations, consultez la section Protection des données dans Simple Storage Service (Amazon S3) dans le Guide de l'utilisateur Amazon Simple Storage Service. Vous contrôlez l'accès à vos données source et pouvez les chiffrer dans Simple Storage Service (Amazon S3). Vous pouvez utiliser Athena pour créer des tables en fonction des ensembles de données chiffrés dans Simple Storage Service (Amazon S3).
- Métadonnées de base de données et de table (schéma) : Athena utilise schema-on-read la technologie, ce qui signifie que les définitions de vos tables sont appliquées à vos données dans Amazon S3 lorsqu'Athena exécute des requêtes. Tous les schémas que vous définissez sont automatiquement enregistrés, sauf si vous les supprimez de manière explicite. Dans Athena, vous pouvez modifier les métadonnées du catalogue de données à l'aide d'instructions DDL. Vous pouvez supprimer les définitions de table et les schémas sans affecter les données sous-jacentes stockées dans Simple Storage Service (Amazon S3). Les métadonnées des bases de données et des tables que vous utilisez dans Athena sont stockées dans le catalogue AWS Glue Data Catalog.

  Vous pouvez définir des politiques d'accès précises aux bases de données et aux tables enregistrées auprès de l'utilisateur AWS Identity and Access Management (AWS Glue Data Catalog IAM). Vous pouvez également chiffrer les métadonnées dans le AWS Glue Data Catalog. Si vous chiffrez les métadonnées, utilisez des autorisations d'accès aux métadonnées chiffrées.
- Résultats de requête et historique des requêtes, y compris les requêtes enregistrées : les résultats de requête sont stockés dans un emplacement dans Simple Storage Service (Amazon S3) que vous pouvez choisir de spécifier globalement ou pour chaque groupe de travail. En l'absence de spécification, Athena utilise l'emplacement par défaut dans chaque cas. Vous contrôlez l'accès aux compartiments Simple Storage Service (Amazon S3) où vous stockez les résultats des requêtes et les requêtes enregistrées. De plus, vous pouvez choisir de chiffrer les résultats de requête que vous stockez dans Simple Storage Service (Amazon S3). Les utilisateurs doivent avoir les autorisations appropriées pour accéder aux emplacements Simple Storage Service (Amazon S3) et déchiffrer les fichiers. Pour plus d'informations, consultez Chiffrement des résultats des requêtes Athena stockées dans Simple Storage Service (Amazon S3) dans ce document.

  Athena conserve l'historique des requêtes pendant 45 jours. Vous pouvez consulter l'historique des requêtes à l'aide des API Athena, dans la console et avec. AWS CLI Pour stocker les requêtes pendant plus de 45 jours, enregistrez-les. Pour protéger l'accès aux requêtes enregistrées, utilisez les groupes de travail dans Athena, en limitant l'accès aux requêtes enregistrées uniquement aux utilisateurs qui sont autorisés à les consulter.

Rubriques

Chiffrement au repos

Vous pouvez exécuter des requêtes dans Amazon Athena sur des données chiffrées dans Simple Storage Service (Amazon S3) dans la même région et dans un nombre limité de régions. Vous pouvez également chiffrer les résultats des requêtes dans Amazon S3 et les données du catalogue de AWS Glue données.

Vous pouvez chiffrer les ressources suivantes dans Athena :

- Les résultats de toutes les requêtes dans Simple Storage Service (Amazon S3), que Athena stocke dans un emplacement appelé emplacement des résultats Simple Storage Service (Amazon S3). Vous pouvez chiffrer les résultats des requêtes stockés dans Simple Storage Service (Amazon S3), que le jeu de données sous-jacent soit chiffré dans Simple Storage Service (Amazon S3) ou non. Pour plus d'informations, veuillez consulter Chiffrement des résultats des requêtes Athena stockées dans Simple Storage Service (Amazon S3).
- Les données du catalogue AWS Glue de données. Pour plus d'informations, veuillez consulter Autorisations sur les métadonnées chiffrées du catalogue de données AWS Glue.

> **Note**
>
> Lorsque vous utilisez Athena pour lire une table chiffrée, Athena utilise les options de chiffrement spécifiées pour les données de la table, et non l'option de chiffrement pour les résultats de la requête. Si des méthodes ou des clés de chiffrement distinctes sont configurées pour les résultats des requêtes et les données des tables, Athena lit les données des tables sans utiliser l'option de chiffrement ni la clé utilisées pour chiffrer ou déchiffrer les résultats de la requête.
>
> Toutefois, si vous utilisez Athena pour insérer des données dans une table contenant des données chiffrées, Athena utilise la configuration de chiffrement spécifiée pour les résultats de la requête afin de chiffrer les données insérées. Par exemple, si vous spécifiez le `CSE_KMS` chiffrement pour les résultats de requête, Athena utilise le même identifiant de

AWS KMS clé que celui que vous avez utilisé pour le chiffrement des résultats de requête pour chiffrer les données de table insérées. `CSE_KMS`

Rubriques

Options de chiffrement Simple Storage Service (Amazon S3) prises en charge

Athena prend en charge les options de chiffrement suivantes pour les jeux de données et les résultats des requêtes dans Simple Storage Service (Amazon S3).

Type de chiffrement	Description	Prise en charge entre régions
SSE-S3	Chiffrement côté serveur (SSE) avec une clé gérée par Simple Storage Service (Amazon S3).	Oui
SSE-KMS	Chiffrement côté serveur (SSE) avec une clé gérée par AWS Key Management Service le client. **Note** Avec ce type de chiffrement, Athena n'exige pas que vous indiquiez que les données sont chiffrées lorsque vous créez une table.	Oui
CSE-KMS	Chiffrement côté client (CSE) avec une clé gérée par AWS KMS le client. Dans Athena, cette option exige que vous utilisiez une instruction `CREATE TABLE` avec une clause	Non

Type de chiffrement	Description	Prise en charge entre régions
	`TBLPROPERTIES` qui spécifie `'has_encrypted_data'='true'`. Pour plus d'informations, consultez Création de tables basées sur des ensembles de données chiffrés dans Simple Storage Service (Amazon S3).	

Pour plus d'informations sur AWS KMS le chiffrement avec Amazon S3, consultez What is AWS Key Management Service and how Amazon Simple Storage Service (Amazon S3) AWS KMS uses dans AWS Key Management Service le manuel du développeur. Pour de plus amples informations sur l'utilisation de SSE-KMS ou CSE-KMS avec Athena, consultez la rubrique Lancement : Amazon Athena prend désormais en charge l'interrogation des données chiffrées à partir du Blog Big Data AWS .

Options non prises en charge

Les options de chiffrement suivantes ne sont pas prises en charge :

- SSE avec les clés fournies par le client (SSE-C).
- Chiffrement côté client à l'aide d'une clé gérée côté client.
- Clés asymétriques.

Pour comparer les options de chiffrement Simple Storage Service (Amazon S3), consultez la section Protection des données à l'aide d'un chiffrement dans le Guide de l'utilisateur Amazon Simple Storage Service.

Outils de chiffrement côté client

Pour le chiffrement côté client, deux outils sont disponibles :

- Client de chiffrement Simple Storage Service (Amazon S3) : cet outil permet de chiffrer les données pour Simple Storage Service (Amazon S3) uniquement et est pris en charge par Athena.
- AWS Encryption SDK— Le SDK peut être utilisé pour chiffrer des données n'importe où, AWS mais il n'est pas directement pris en charge par Athena.

Ces outils ne sont pas compatibles, et les données chiffrées à l'aide d'un outil ne peuvent pas être déchiffrées par un autre outil. Athena ne prend en charge directement que le client de chiffrement Simple Storage Service (Amazon S3). Si vous utilisez le kit SDK pour chiffrer vos données, vous pouvez exécuter des requêtes depuis Athena, mais les données sont renvoyées sous forme de texte chiffré.

Si vous souhaitez utiliser Athena pour interroger des données qui ont été chiffrées à l'aide du kit SDK de chiffrement AWS , vous devez télécharger et déchiffrer vos données, puis les chiffrer à nouveau à l'aide du client de chiffrement Simple Storage Service (Amazon S3).

Autorisations pour les données chiffrées dans Simple Storage Service (Amazon S3)

Selon le type de chiffrement que vous utilisez dans Simple Storage Service (Amazon S3), vous devrez peut-être ajouter des autorisations, également appelées actions « Autoriser », à vos politiques utilisées dans Athena :

- SSE-S3 : si vous utilisez SSE-S3 pour le chiffrement, les utilisateurs d'Athena n'ont besoin d'aucune autorisation supplémentaire dans leurs politiques. Il suffit de disposer des autorisations Simple Storage Service (Amazon S3) appropriées pour l'emplacement Simple Storage Service (Amazon S3) approprié et pour les actions Athena. Pour plus d'informations sur les politiques qui accordent les autorisations Athena et Amazon S3 appropriées, consultez AWS politiques gérées pour Amazon Athena et Accès à Amazon S3 depuis Athena.
- AWS KMS— Si vous l'utilisez AWS KMS pour le chiffrement, les utilisateurs d'Athena doivent être autorisés à effectuer des AWS KMS actions spécifiques en plus des autorisations Athena et Amazon S3. Vous autorisez ces actions en modifiant la politique clé pour les CMK gérées par le AWS KMS client qui sont utilisées pour chiffrer les données dans Amazon S3. Pour ajouter des utilisateurs clés aux politiques AWS KMS clés appropriées, vous pouvez utiliser la AWS KMS console à l'adresse https://console.aws.amazon.com/kms. Pour plus d'informations sur la façon d'ajouter un utilisateur à une politique AWS KMS clé, voir Autoriser les utilisateurs clés à utiliser la clé CMK dans le guide du AWS Key Management Service développeur.

> **Note**
>
> Les administrateurs de politiques de clés avancées peuvent adapter les politiques de clés. `kms:Decrypt` est l'action minimale autorisée pour qu'un utilisateur d'Athena puisse travailler avec un jeu de données chiffrées. Pour utiliser des résultats de requête chiffrés, les actions minimales autorisées sont `kms:GenerateDataKey` et `kms:Decrypt`.

Lorsque vous utilisez Athena pour interroger des ensembles de données dans Amazon S3 contenant un grand nombre d'objets chiffrés AWS KMS, cela AWS KMS peut limiter les résultats des requêtes. Ceci est plus probable lorsqu'il y a un grand nombre de petits objets. Athena collecte les requêtes de nouvelles tentatives, mais une erreur de limitation peut encore se produire. Si vous travaillez avec un grand nombre d'objets chiffrés et que vous rencontrez ce problème, une option consiste à activer les clés de compartiment Simple Storage Service (Amazon S3) pour réduire le nombre d'appels vers KMS. Pour plus d'informations, consultez la rubrique Réduction des coûts de SSE-KMS grâce aux clés de compartiment Simple Storage Service (Amazon S3) dans le Guide de l'utilisateur d'Amazon Simple Storage Service. Une autre option consiste à augmenter vos quotas de service pour AWS KMS. Pour de plus amples informations, veuillez consulter Quotas dans le Guide du développeur AWS Key Management Service .

Pour obtenir des informations de dépannage sur les autorisations lors de l'utilisation de Simple Storage Service (Amazon S3) avec Athena, voir la section Autorisations de la rubrique Résolution des problèmes dans Athena.

Autorisations sur les métadonnées chiffrées du catalogue de données AWS Glue

Si vous cryptez des métadonnées dans le AWS Glue Data Catalog, vous devez ajouter `"kms:GenerateDataKey"` des `"kms:Encrypt"` actions et ajouter des actions aux politiques que vous utilisez pour accéder à Athena. `"kms:Decrypt"` Pour plus d'informations, veuillez consulter Accès depuis Athena aux métadonnées cryptées dans le AWS Glue Data Catalog.

Chiffrement des résultats des requêtes Athena stockées dans Simple Storage Service (Amazon S3)

Vous pouvez configurer le chiffrement des résultats des requêtes à l'aide de la console Athena ou lors de l'utilisation de JDBC ou ODBC. Les groupes de travail vous permettent d'appliquer le chiffrement des résultats de la requête.

Dans la console, vous pouvez configurer le paramètre de chiffrement des résultats des requêtes de deux manières :

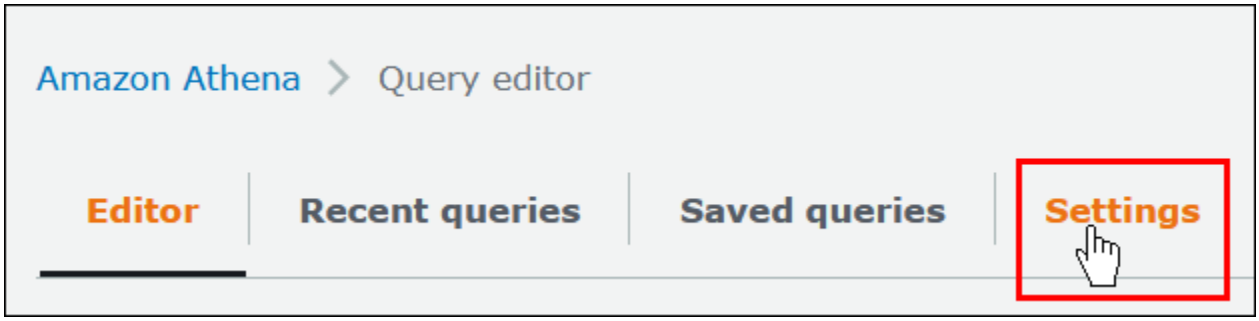
- Paramètres côté client : lorsque vous utilisez Settings (Paramètres) de la console ou les opérations de l'API pour indiquer que vous souhaitez chiffrer les résultats des requêtes, vous utilisez les paramètres côté client. Les paramètres côté client incluent l'emplacement des résultats de requête et le chiffrement. Si vous les spécifiez, ils sont utilisés, sauf s'ils sont remplacés par les paramètres du groupe de travail.

- Paramètres de groupe de travail : lorsque vous créez ou modifiez un groupe de travail et sélectionnez le champ Override client-side settings (Remplacer les paramètres côté client), toutes les requêtes s'exécutant dans ce groupe de travail utilisent les paramètres du groupe de travail et d'emplacement des résultats de requête. Pour plus d'informations, consultez Les paramètres du groupe de travail remplacent les paramètres côté client.

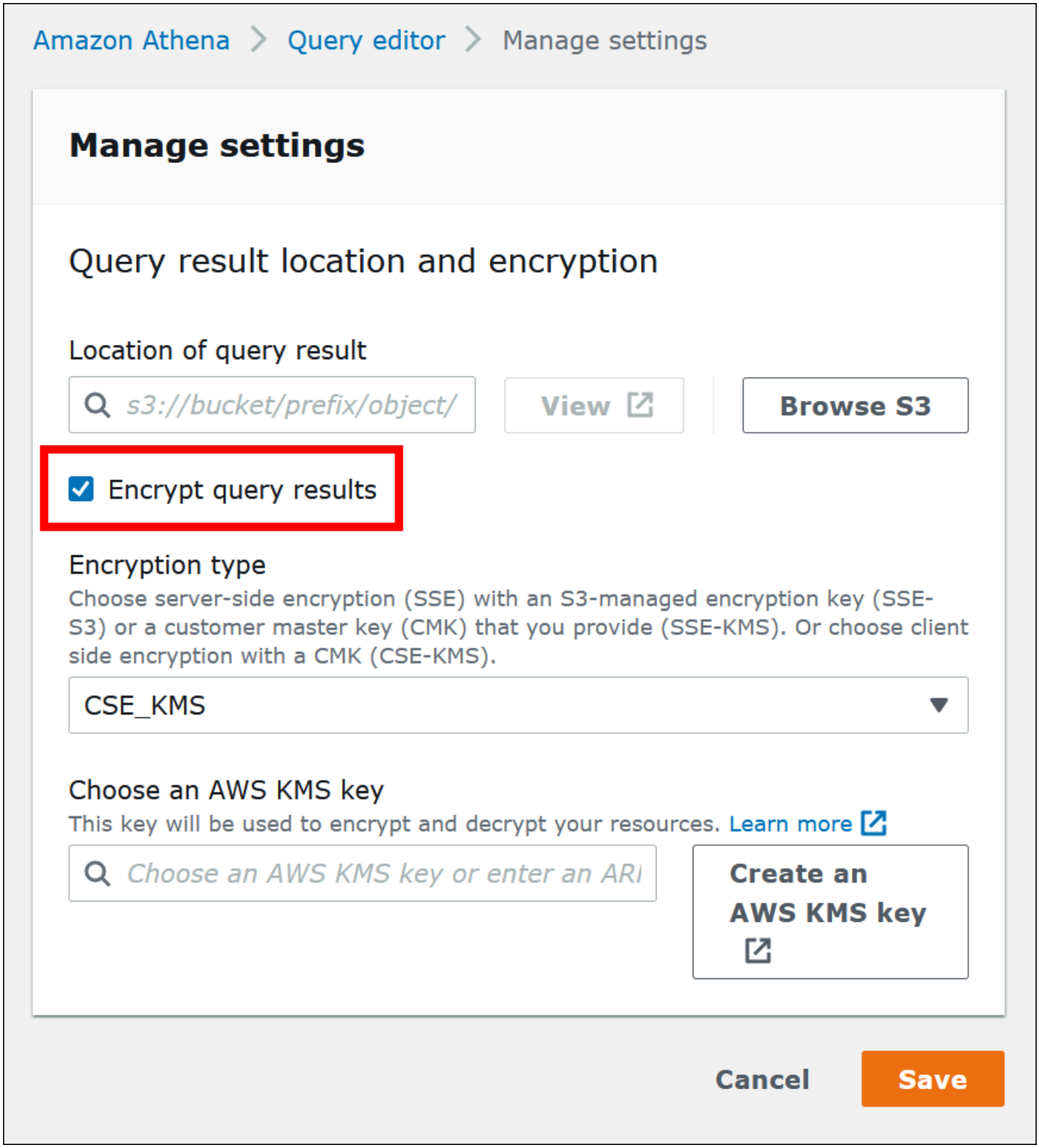
Pour chiffrer les résultats des requêtes stockées dans Simple Storage Service (Amazon S3) à l'aide de la console

> **Important**
>
> Si le champ Override Client-Side Setting (Remplacer les paramètres côté client) est sélectionné pour votre groupe de travail, toutes les requêtes dans le groupe de travail utilisent alors les paramètres du groupe de travail. La configuration du chiffrement et l'emplacement des résultats de requête définis dans l'onglet Settings (Paramètres) dans la console Athena par les opérations d'API et les pilotes JDBC et ODBC ne sont pas utilisés. Pour plus d'informations, consultez Les paramètres du groupe de travail remplacent les paramètres côté client.

1. Dans la console Athena, choisissez Settings (Paramètres).



2. Choisissez Gérer.
3. Pour Location of query result (Emplacement du résultat de la requête), saisissez ou choisissez un chemin Simple Storage Service (Amazon S3). Il s'agit de l'emplacement Simple Storage Service (Amazon S3) où les résultats de votre requête seront stockés.
4. Choisissez Encrypt query results.



5. Pour Encryption type, choisissez CSE, KMS, SSE-KMS ou SSE-S3. Parmi ces trois solutions, CSE-KMS offre le niveau de chiffrement le plus élevé et SSE-S3 le plus bas.
6. Si vous avez choisi SSE-KMS ou CSE-KMS, spécifiez une clé. AWS KMS

- Pour Choisir une AWS KMS clé, si votre compte a accès à une clé gérée par AWS KMS le client (CMK) existante, choisissez son alias ou entrez un ARN de AWS KMS clé.
- Si votre compte n'a pas accès à une clé gérée par le client (CMK) existante, choisissez Créer une AWS KMS clé, puis ouvrez la AWS KMS console. Pour plus d'informations, consultez Création des clés dans le Guide du développeur AWS Key Management Service .

> **Note**
>
> Athena ne prend en charge que les clés symétriques pour la lecture et l'écriture de données.

7. Revenez à la console Athena et choisissez la clé que vous avez créée par alias ou ARN.
8. Choisissez Enregistrer.

Chiffrement des résultats des requêtes Athena lors de l'utilisation de JDBC ou ODBC

Si vous vous connectez à l'aide du pilote JDBC ou ODBC, vous configurez les options du pilote pour spécifier le type de chiffrement à utiliser et l'emplacement du répertoire Simple Storage Service (Amazon S3) intermédiaire. Pour configurer le pilote JDBC pour chiffrer vos résultats de requête en utilisant l'un des protocoles de chiffrement pris en charge par Athena, consultez Connexion à Amazon Athena avec les pilotes ODBC et JDBC.

Création de tables basées sur des ensembles de données chiffrés dans Simple Storage Service (Amazon S3)

Lorsque vous créez une table, indiquez à Athena qu'un jeu de données est chiffré dans Simple Storage Service (Amazon S3). Ceci n'est pas nécessaire lors de l'utilisation de SSE-KMS. Pour le SSE-S3 et le AWS KMS chiffrement, Athena détermine comment déchiffrer l'ensemble de données et créer la table, de sorte que vous n'avez pas besoin de fournir d'informations clés.

Les utilisateurs qui exécutent des requêtes, y compris l'utilisateur qui crée la table, doivent avoir les autorisations décrites précédemment dans cette rubrique.

> **Important**
>
> Si vous utilisez Amazon EMR avec EMRFS pour télécharger des fichiers Parquet chiffrés, vous devez désactiver les chargements partitionnés en définissant

`fs.s3n.multipart.uploads.enabled` sur `false`. Sinon, Athena n'est pas en mesure de déterminer la longueur des fichiers Parquet et une erreur HIVE_CANNOT_OPEN_SPLIT se produit. Pour plus d'informations, consultez Configuration d'un chargement partitionné pour Simple Storage Service (Amazon S3) dans le Guide de gestion Amazon EMR.

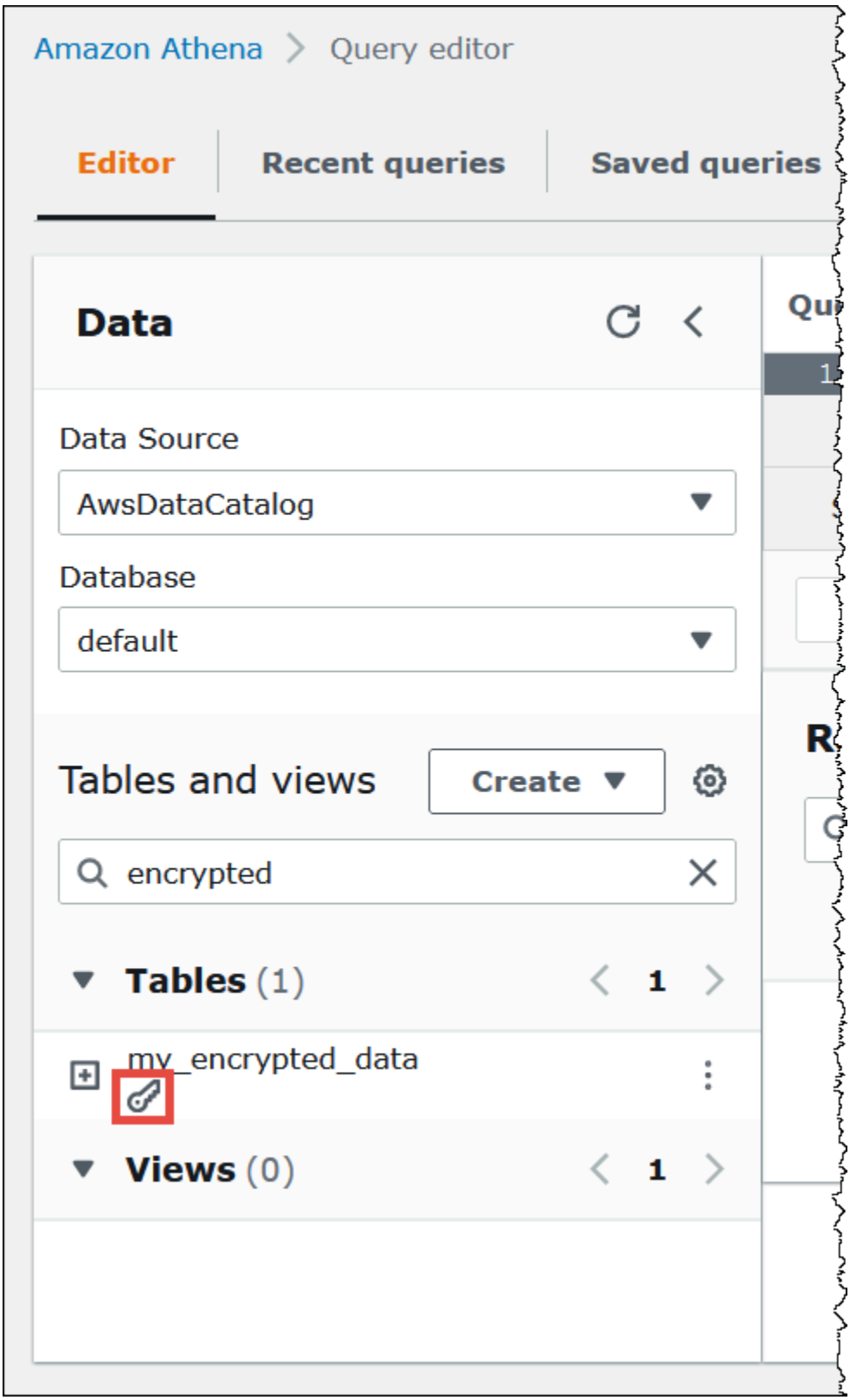
Pour indiquer que le jeu de données est chiffré dans Simple Storage Service (Amazon S3), suivez l'une des étapes ci-dessous. Cette étape n'est pas nécessaire si SSE-KMS est utilisé.

- Dans une instruction CREATE TABLE (CRÉER UNE TABLE), utilisez une clause `TBLPROPERTIES` qui spécifie `'has_encrypted_data'='true'`, comme dans l'exemple ci-dessous.

```
CREATE EXTERNAL TABLE 'my_encrypted_data' (
   `n_nationkey` int,
   `n_name` string,
   `n_regionkey` int,
   `n_comment` string)
ROW FORMAT SERDE
   'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
   'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
LOCATION
   's3://DOC-EXAMPLE-BUCKET/folder_with_my_encrypted_data/'
TBLPROPERTIES (
   'has_encrypted_data'='true')
```

- Utilisez le pilote JDBC et définissez la valeur `TBLPROPERTIES` comme indiqué dans l'exemple précédent lorsque vous utilisez `statement.executeQuery()` pour exécuter l'instruction CREATE TABLE (CRÉER UNE TABLE).
- Lorsque vous utilisez la console Athena pour créer une table à l'aide d'un formulaire et que vous spécifiez l'emplacement de la table, sélectionnez l'option Encrypted data set (Jeu de données chiffrées).

Dans la liste des tables de la console Athena, les tables chiffrées affichent une icône en forme de clé.


Amazon Athena
Query editor
Editor
Recent queries
Saved queries
Data
Data Source
AwsDataCatalog
Database
default
Tables and views
Create
encrypted
Tables (1)
1
my_encrypted_data
Views (0)
1

Chiffrement en transit

Outre le chiffrement de données au repos dans Simple Storage Service (Amazon S3), Amazon Athena utilise le protocole de chiffrement TLS (Transport Layer Security, Sécurité de la couche de transport) pour le chiffrement des données en transit entre Athena et Simple Storage Service (Amazon S3), et entre Athena et les applications clientes y accédant.

Vous devez autoriser uniquement les connexions chiffrées sur HTTPS (TLS) avec `aws:SecureTransport condition` sur les politiques IAM de compartiment Simple Storage Service (Amazon S3).

Les résultats de requête qui diffusent en continu vers les clients JDBC ou ODBC sont chiffrés à l'aide du protocole TLS. Pour de plus amples informations sur les dernières versions des pilotes JDBC et ODBC et leur documentation, veuillez consulter Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

Pour les connecteurs de source de données fédérés Athena, la prise en charge du chiffrement en transit à l'aide du protocole TLS dépend du connecteur individuel. Pour plus d'informations, consultez la documentation relative aux connecteurs de source de données individuels.

Gestion des clés

Amazon Athena prend en charge AWS Key Management Service (AWS KMS) pour chiffrer les ensembles de données dans Amazon S3 et les résultats des requêtes Athena. AWS KMS utilise des clés gérées par le client (CMK) pour chiffrer vos objets Amazon S3 et s'appuie sur le chiffrement des enveloppes.

Dans AWS KMS, vous pouvez effectuer les actions suivantes :

- Créer des clés
- Importer vos propres éléments de clé pour les nouvelles clés CMK

> **Note**
>
> Athena ne prend en charge que les clés symétriques pour la lecture et l'écriture de données.

Pour de plus amples informations, consultez la rubrique Présentation de AWS Key Management Service dans le Guide du développeur AWS Key Management Service et Comment Amazon Simple

Storage Service utilise le service AWS KMS. Pour afficher les clés de votre compte qui AWS crée et gère pour vous, dans le volet de navigation, choisissez les clés AWS gérées.

Si vous téléchargez ou accédez à des objets chiffrés par SSE-KMS, utilisez AWS Signature Version 4 pour plus de sécurité. Pour plus d'informations, consultez la section Spécification de la version de la signature dans l'authentification des requêtes dans le Guide de l'utilisateur d'Amazon Simple Storage Service.

Si vos charges de travail Athena chiffrent une grande quantité de données, vous pouvez utiliser des clés de compartiment Simple Storage Service (Amazon S3) pour réduire les coûts. Pour plus d'informations, consultez la rubrique Réduction des coûts de SSE-KMS grâce aux clés de compartiment Simple Storage Service (Amazon S3) dans le Guide de l'utilisateur d'Amazon Simple Storage Service.

Confidentialité du trafic inter-réseau

Le trafic est protégé à la fois entre Athena et les applications sur site et entre Athena et Simple Storage Service (Amazon S3). Le trafic entre Athena et d'autres services, tels que AWS Glue et AWS Key Management Service, utilise le protocole HTTPS par défaut.

- Pour le trafic entre et les clients sur site et les applications, les résultats des requêtes diffusés vers JDBC ou ODBC sont chiffrés à l'aide du protocole TLS (Transport Layer Security, Sécurité de la couche de transport).

  Vous pouvez utiliser l'une des options de connectivité entre votre réseau privé et AWS :

  - Une connexion VPN de site à site. AWS VPN Pour plus d'informations, consultez la rubrique Qu'est-ce que Site-to-Site VPN (AWS VPN site à site) du Guide de l'utilisateur AWS Site-to-Site VPN .
  - Une AWS Direct Connect connexion. Pour plus d'informations, consultez Présentation de AWS Direct Connect dans le Guide de l'utilisateur AWS Direct Connect .
- Pour le trafic entre Athena et les compartiments Simple Storage Service (Amazon S3), le protocole TLS (Transport Layer Security, Sécurité de la couche de transport) chiffre les objets qui se déplacent entre Athena et Simple Storage Service (Amazon S3), et entre Athena et les applications y accédant, vous devez autoriser uniquement les connexions chiffrées sur HTTPS (TLS) à l'aide de la `aws:SecureTransport condition` sur les politiques IAM de compartiment Simple Storage Service (Amazon S3). Bien qu'Athena utilise actuellement le point de terminaison public pour accéder aux données des compartiments Simple Storage Service (Amazon S3), cela ne signifie

pas que les données passent par l'Internet public. Tout le trafic entre Athena et Amazon S3 est acheminé sur le AWS réseau et chiffré à l'aide du protocole TLS.

- Programmes de conformité : Amazon Athena est conforme à de nombreux programmes de AWS conformité, notamment SOC, PCI, FedRAMP, etc. Pour de plus amples informations, veuillez consulter les Services AWS concernés par le programme de conformité.

Gestion des identités et des accès dans Athena

Amazon Athena utilise des politiques AWS Identity and Access Management (IAM) pour restreindre l'accès aux opérations Athena. Pour une liste complète des autorisations pour Athena, consultez la rubrique Actions, ressources et clés de condition pour Amazon Athena de la section Référence pour l'autorisation des services.

Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Parmi les autorisations requises pour exécuter les requêtes Athena figurent les suivantes :

- Emplacements Simple Storage Service (Amazon S3) où sont stockées les données sous-jacentes à interroger. Pour plus d'informations, consultez la section Identity and access management dans Amazon S3 du Guide de l'utilisateur Amazon Simple Storage Service.
- Les métadonnées et les ressources que vous y stockez AWS Glue Data Catalog, telles que les bases de données et les tables, y compris les actions supplémentaires pour les métadonnées chiffrées. Pour plus d'informations, consultez la rubrique Configuration des autorisations IAM pour AWS Glue et Configuration du chiffrement dans AWS Glue du Guide du développeur AWS Glue .
- Actions API dans Athena. Pour obtenir une liste des actions API dans Athena, consultez la rubrique Actions de la section Référence API d'Amazon Athena.

Les rubriques suivantes fournissent de plus amples informations sur les autorisations pour des zones spécifiques d'Athena.

Rubriques

- AWS politiques gérées pour Amazon Athena
- Accès via les connexions JDBC et ODBC
- Accès à Amazon S3 depuis Athena
- Accès inter-comptes aux compartiments Simple Storage Service (Amazon S3) dans Athena

- Accès détaillé aux bases de données et aux tables du AWS Glue Data Catalog
- Accès entre comptes aux catalogues de données AWS Glue
- Accès depuis Athena aux métadonnées cryptées dans le AWS Glue Data Catalog
- Accès aux groupes de travail et identifications
- Autorisation d'accès aux instructions préparées
- Utilisation d'Athéna avec CalledVia des touches contextuelles
- Autorisation d'accès à un connecteur de données Athena pour un métastore Hive externe
- Autorisation d'accès des fonctions Lambda aux métastores Hive externes
- Exemple de politiques d'autorisation IAM pour autoriser la requête fédérée Athena
- Exemple de politiques d'autorisations IAM pour autoriser les fonctions définies par l'utilisateur (UDF) Amazon Athena
- Autorisation d'accès pour ML avec Athena
- Activation de l'accès fédéré à l'API Athena

AWS politiques gérées pour Amazon Athena

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle politique Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez la section Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

Aspects à prendre en compte lors de l'utilisation de politiques gérées avec Athena

Les politiques gérées sont faciles à utiliser et sont automatiquement mises à jour avec les actions requises, à mesure que le service évolue. Lorsque vous utilisez des politiques gérées avec Athena, gardez à l'esprit les points suivants :

- Pour autoriser ou refuser les actions du service Amazon Athena pour vous-même ou pour d'autres utilisateurs avec AWS Identity and Access Management (IAM), vous attachez des politiques basées sur l'identité aux principaux, tels que les utilisateurs ou les groupes.
- Chaque politique basée sur une identité se compose d'instructions qui définissent les actions qui sont autorisées ou refusées. Pour plus d'informations et des step-by-step instructions relatives à l'attachement d'une politique à un utilisateur, consultez la section Attacher des politiques gérées dans le guide de l'utilisateur IAM. Pour obtenir une liste des actions, consultez la Référence d'API Amazon Athena.
- Les politiques basées sur l'identité, gérées par le client et en ligne, vous permettent de spécifier des actions Athena plus détaillées au sein d'une politique pour affiner l'accès. Nous vous recommandons d'utiliser la politique `AmazonAthenaFullAccess` comme point de départ, puis d'autoriser ou de refuser des actions spécifiques figurant dans le manuel Référence d'API Amazon Athena. Pour de plus amples informations sur les politiques en ligne, consultez Politiques gérées et politiques en ligne dans le Guide de l'utilisateur IAM.
- Si vous avez également des principaux qui se connectent en utilisant JDBC, vous devez fournir les informations d'identification du pilote JDBC à votre application. Pour plus d’informations, consultez Accès via les connexions JDBC et ODBC.
- Si vous avez chiffré le catalogue de AWS Glue données, vous devez spécifier des actions supplémentaires dans les politiques IAM basées sur l'identité pour Athena. Pour plus d’informations, consultez Accès depuis Athena aux métadonnées cryptées dans le AWS Glue Data Catalog.
- Si vous créez et utilisez des groupes de travail, assurez-vous que vos politiques prévoient un accès pertinent aux actions du groupe de travail. Pour plus d'informations, consultez the section called “ Politiques IAM pour l'accès aux groupes de travail” et the section called “Exemples de politiques de groupe de travail”.

AWS politique gérée : AmazonAthenaFullAccess

La politique gérée par `AmazonAthenaFullAccess` accorde un accès complet à Athena.

Pour activer l’accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

  Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center .

- Utilisateurs gérés dans IAM par un fournisseur d'identité :

  Créez un rôle pour la fédération d'identité. Pour plus d'informations, voir la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) du Guide de l'utilisateur IAM.

- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM du Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

Groupes d'autorisations

La politique est `AmazonAthenaFullAccess` regroupée dans les ensembles d'autorisations suivants.

- **`athena`** : permet aux principaux d'accéder aux ressources Athena.
- **`glue`**— Permet aux principaux d'accéder aux AWS Glue bases de données, aux tables et aux partitions. Cela est nécessaire pour que le directeur puisse utiliser le AWS Glue Data Catalog avec Athéna.
- **`s3`** : permet au principal d'écrire et de lire les résultats des requêtes à partir de Simple Storage Service (Amazon S3), de lire les exemples de données Athena disponibles publiquement qui résident dans Simple Storage Service (Amazon S3) et de répertorier les compartiments. Ceci est nécessaire pour que le principal puisse utiliser Athena pour travailler avec Simple Storage Service (Amazon S3).
- **`sns`** : permet aux principaux de répertorier les rubriques Amazon SNS et d'obtenir les attributs de rubrique. Cela permet aux principaux d'utiliser les rubriques Amazon SNS avec Athena à des fins de surveillance et d'alerte.
- **`cloudwatch`**— Permet aux principaux de créer, de lire et de supprimer des CloudWatch alarmes. Pour plus d'informations, consultez Contrôle des coûts et surveillance des requêtes à l'aide de CloudWatch métriques et d'événements.

- **`lakeformation`** : permet aux principaux de demander des informations d'identification temporaires pour accéder aux données dans un emplacement de lac de données enregistré auprès de Lake Formation. Pour plus d'informations, consultez la rubrique Contrôle d'accès aux données sous-jacentes du Guide du développeur AWS Lake Formation.
- **`datazone`**— Permet aux principaux de répertorier les DataZone projets, domaines et environnements Amazon. Pour plus d'informations sur l'utilisation DataZone dans Athena, consultez. Utilisation d'Amazon DataZone dans Athena
- **`pricing`**— Donne accès à AWS Billing and Cost Management. Pour plus d'informations, consultez GetProductsla référence de AWS Billing and Cost Management l'API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "BaseAthenaPermissions",
            "Effect": "Allow",
            "Action": [
                "athena:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseGluePermissions",
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
```

```
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:StartColumnStatisticsTaskRun",
                "glue:GetColumnStatisticsTaskRun",
                "glue:GetColumnStatisticsTaskRuns",
                "glue:GetCatalogImportStatus"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseQueryResultsPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": [
                "arn:aws:s3:::aws-athena-query-results-*"
            ]
        },
        {
            "Sid": "BaseAthenaExamplesPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::athena-examples*"
            ]
        },
```

```
        {
            "Sid": "BaseS3BucketPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseSNSPermissions",
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics",
                "sns:GetTopicAttributes"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseCloudWatchPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:GetMetricData"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseLakeFormationPermissions",
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
```

```
            ]
        },
        {
            "Sid": "BaseDataZonePermissions",
            "Effect": "Allow",
            "Action": [
                "datazone:ListDomains",
                "datazone:ListProjects",
                "datazone:ListAccountEnvironments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BasePricingPermissions",
            "Effect": "Allow",
            "Action": [
                "pricing:GetProducts"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS politique gérée : AWSQuicksightAthenaAccess

`AWSQuicksightAthenaAccess`donne accès aux actions dont Amazon a QuickSight besoin pour s'intégrer à Athena. Vous pouvez associer la politique `AWSQuicksightAthenaAccess` à vos identités IAM. N'associez cette politique qu'aux principaux utilisateurs d'Amazon QuickSight avec Athena. Cette politique comprend certaines actions pour Athena qui sont soit obsolètes et non incluses dans l'API publique actuelle, soit utilisées uniquement avec les pilotes JDBC et ODBC.

Groupes d'autorisations

La politique est `AWSQuicksightAthenaAccess` regroupée dans les ensembles d'autorisations suivants.

- **`athena`** : permet au principal d'exécuter des requêtes sur les ressources Athena.

- **`glue`**— Permet aux principaux d'accéder aux AWS Glue bases de données, aux tables et aux partitions. Cela est nécessaire pour que le directeur puisse utiliser le AWS Glue Data Catalog avec Athéna.
- **`s3`** : permet au principal d'écrire et de lire les résultats des requêtes depuis Simple Storage Service (Amazon S3).
- **`lakeformation`** – Permet aux principaux de demander des informations d'identification temporaires pour accéder aux données dans un emplacement de lac de données enregistré auprès de Lake Formation. Pour plus d'informations, consultez la rubrique Contrôle d'accès aux données sous-jacentes du Guide du développeur AWS Lake Formation.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:BatchGetQueryExecution",
                "athena:GetQueryExecution",
                "athena:GetQueryResults",
                "athena:GetQueryResultsStream",
                "athena:ListQueryExecutions",
                "athena:StartQueryExecution",
                "athena:StopQueryExecution",
                "athena:ListWorkGroups",
                "athena:ListEngineVersions",
                "athena:GetWorkGroup",
                "athena:GetDataCatalog",
                "athena:GetDatabase",
                "athena:GetTableMetadata",
                "athena:ListDataCatalogs",
                "athena:ListDatabases",
                "athena:ListTableMetadata"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": [
                "arn:aws:s3:::aws-athena-query-results-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Athena met à jour ses politiques gérées AWS

Consultez les détails des mises à jour des politiques AWS gérées pour Athena depuis que ce service a commencé à suivre ces modifications.

Modification	Description	Date
AmazonAthenaFullAccess – Mise à jour de la politique existante	Permet à Athena d'utiliser l' AWS Glue `GetCatalogImportStatus` API documentée publiquement pour récupérer le statut d'importation du catalogue.	18 juin 2024
AmazonAthenaFullAccess – Mise à jour de la politique existante	Les `datazone:ListAccountEnvironments` autorisations `datazone:ListDomains` `datazone:ListProjects` , et ont été ajoutées pour permettre aux utilisateurs d'Athena de travailler avec des DataZone domaines, des projets et des environnements Amazon. Pour plus d'informations, consultez Utilisation d'Amazon DataZone dans Athena.	3 janvier 2024
AmazonAthenaFullAccess – Mise à jour de la politique existante	Les `glue:GetColumnStatisticsTaskRuns` autorisations `glue:StartColumnStatisticsT`	3 janvier 2024

Modification	Description	Date
	`askRun glue:GetC olumnStatisticsTas kRun` , et ont été ajoutées pour donner à Athena le droit d'appeler pour récupérer les statistiques relatives AWS Glue à la fonction d'optimisation basée sur les coûts. Pour plus d'informations, consultez Utilisation de l'optimiseur basé sur les coûts.	
AmazonAthenaFullAccess – Mise à jour de la politique existante	Athena a ajouté `pricing:GetProducts` pour donner accès à AWS Billing and Cost Management. Pour plus d'informations, consultez GetProductsla référence de AWS Billing and Cost Management l'API.	25 janvier 2023
AmazonAthenaFullAccess – Mise à jour de la politique existante	Athena a été ajoutée `cloudwatch:GetMetricData` pour récupérer les valeurs CloudWatch métriques. Pour plus d'informations, consultez GetMetricDatale Amazon CloudWatch API Reference.	14 novembre 2022
AmazonAthenaFullAccesset AWSQuicksightAthenaAccess— Mises à jour des politiques existantes	Athena a ajouté `s3:PutBucketPublicAccessBlock` pour permettre le blocage de l'accès public aux compartiments créés par Athena.	7 juillet 2021
Athena a commencé à suivre les modifications	Athena a commencé à suivre les modifications apportées à ses politiques AWS gérées.	7 juillet 2021

Accès via les connexions JDBC et ODBC

Pour accéder à des ressources, telles qu'Athena Services AWS et les compartiments Amazon S3, fournissez les informations d'identification du pilote JDBC ou ODBC à votre application. Si vous utilisez le pilote JDBC ou ODBC, assurez-vous que la politique d'autorisations IAM comprend toutes les actions répertoriées dans AWS politique gérée : AWSQuicksightAthenaAccess.

Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Méthodes d'authentification

Les pilotes Athena JDBC et ODBC prennent en charge l'authentification basée sur SAML 2.0, y compris les fournisseurs d'identité suivants :

- Active Directory Federation Services (AD FS)
- Azure Active Directory (AD)
- Okta
- PingFederate

Pour plus d’informations, consultez les guides d’installation et de configuration des pilotes respectifs, téléchargeables au format PDF à partir des pages consacrées aux pilotes JDBC et ODBC. Pour d'autres informations connexes, consultez les rubriques suivantes :

- Activation de l'accès fédéré à l'API Athena
- Utilisation de Lake Formation et des pilotes JDBC et ODBC d'Athena pour l'accès fédéré à Athena
- Configuration de l'authentification unique à l'aide d'ODBC, SAML 2.0 et du fournisseur d'identité Okta

Pour de plus amples informations sur les dernières versions des pilotes JDBC et ODBC et leur documentation, veuillez consulter Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

Accès à Amazon S3 depuis Athena

Vous pouvez accorder l'accès aux emplacements Simple Storage Service (Amazon S3) en utilisant des politiques basées sur l'identité, des politiques de ressources de compartiment, des politiques de

point d'accès, ou toute combinaison de ces politiques. Lorsque les acteurs interagissent avec Athena, leurs autorisations passent par Athena pour déterminer ce à quoi Athena peut accéder. Cela signifie que les utilisateurs doivent être autorisés à accéder aux compartiments Amazon S3 pour pouvoir les interroger avec Athena.

Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Notez que les demandes adressées à Amazon S3 proviennent d'une adresse IPv4 privée pour Athena, et non de l'adresse IP source spécifiée dans. `aws:SourceIp` Pour cette raison, vous ne pouvez pas utiliser cette `aws:SourceIp` condition pour refuser l'accès aux actions Amazon S3 dans le cadre d'une politique IAM donnée. Vous ne pouvez pas non plus restreindre ou autoriser l'accès aux ressources Amazon S3 en fonction des clés de `aws:SourceVpce` condition `aws:SourceVpc` ou.

> **Note**
>
> Les groupes de travail Athena qui utilisent l'authentification IAM Identity Center nécessitent que les autorisations d'accès S3 soient configurées pour utiliser la propagation d'identité approuvée. Pour plus d'informations, consultez la rubrique S3 Access Grants and directory identities dans le Guide de l'utilisateur Amazon Simple Storage Service.

Rubriques

Contrôle de l'accès aux compartiments Amazon S3 à l'aide de politiques basées sur l'identité

Les politiques basées sur une identité sont attachées à un utilisateur, un groupe ou un rôle IAM. Ces politiques vous permettent de spécifier ce que peut faire cette identité (ses autorisations). Vous pouvez utiliser des politiques basées sur l'identité pour contrôler l'accès à vos compartiments Amazon S3.

La politique basée sur l'identité suivante autorise `Read` et autorise `Write` l'accès aux objets d'un compartiment Amazon S3 spécifique. Pour appliquer cette politique, remplacez le *`texte de l'espace réservé en italique`* par vos propres valeurs.

```
{
"Version": "2012-10-17",
"Statement":
    [
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource":
                ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"]
        },
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource":
                ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"]
        }
    ]
}
```

Contrôle de l'accès aux compartiments Amazon S3 à l'aide de politiques relatives aux ressources des compartiments

Vous pouvez utiliser les politiques relatives aux compartiments Amazon S3 pour sécuriser l'accès aux objets de vos compartiments afin que seuls les utilisateurs disposant des autorisations appropriées puissent y accéder. Pour obtenir des conseils sur la création de votre politique Amazon S3, consultez la section Ajouter une politique de compartiment à l'aide de la console Amazon S3 dans le guide de l'utilisateur Amazon S3.

L'exemple de politique d'autorisation suivant limite un utilisateur à la lecture d'objets dotés de la clé et de la valeur de `environment: production` balise. L'exemple de politique utilise la clé de `s3:ExistingObjectTag` condition pour spécifier la clé et la valeur de la balise.

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

```
    [
        {
            "Principal":{"AWS":"arn:aws:iam::111122223333:role/JohnDoe"
        },
            "Effect":"Allow",
            "Action": [ "s3:GetObject", "s3:GetObjectVersion" ],
            "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition":
                {
                    "StringEquals":{"s3:ExistingObjectTag/environment":"production"
                }
    ]
}
```

Pour plus d'exemples de politiques relatives aux compartiments, consultez la section Exemples de politiques relatives aux compartiments Amazon S3 dans le guide de l'utilisateur Amazon S3.

Points d'accès Amazon S3, alias de point d'accès et politiques relatives aux points d'accès

Si vous disposez d'un jeu de données partagé dans un compartiment Simple Storage Service (Amazon S3), il peut s'avérer difficile de maintenir une politique unique pour le compartiment qui gère l'accès à des centaines de cas d'utilisation.

Les points d'accès aux compartiments Simple Storage Service (Amazon S3) permettent de résoudre ce problème. Un compartiment peut avoir plusieurs points d'accès, chacun avec une politique qui contrôle l'accès au compartiment d'une manière différente.

Pour chaque point d'accès que vous créez, Simple Storage Service (Amazon S3) génère un alias qui représente le point d'accès. L'alias étant au format du nom du compartiment Simple Storage Service (Amazon S3), vous pouvez l'utiliser dans la clause `LOCATION` de vos instructions `CREATE TABLE` dans Athena. L'accès d'Athena au compartiment est alors contrôlé par la politique du point d'accès que l'alias représente.

Pour plus d'informations, voir Emplacement de table dans Simple Storage Service (Amazon S3) et Utilisation des points d'accès du Guide de l'utilisateur Simple Storage Service (Amazon S3).

Utilisation des touches CalledVia contextuelles

Pour plus de sécurité, vous pouvez utiliser la clé contextuelle de condition globale aws:CalledVia. La clé `aws:CalledVia` contient une liste ordonnée des services de la chaîne ayant effectué des demandes pour le compte du principal. En spécifiant le nom principal du service Athena

`athena.amazonaws.com` pour la clé contextuelle `aws:CalledVia`, vous pouvez limiter les requêtes à celles effectuées à partir d'Athena. Pour plus d'informations, consultez Utilisation d'Athéna avec CalledVia des touches contextuelles.

Ressources supplémentaires

Pour des informations détaillées et des exemples sur la manière d'accorder l'accès à Simple Storage Service (Amazon S3), consultez les ressources suivantes :

- Exemples de procédures : gestion de l'accès dans le Guide de l'utilisateur Simple Storage Service (Amazon S3).
- Comment puis-je fournir un accès entre comptes aux objets qui se trouvent dans des compartiments Amazon S3 ? dans le AWS Knowledge Center.
- Accès inter-comptes aux compartiments Simple Storage Service (Amazon S3) dans Athena.

Accès inter-comptes aux compartiments Simple Storage Service (Amazon S3) dans Athena

Un scénario courant d'Amazon Athena consiste à accorder l'accès à des utilisateurs dans un compte différent de celui du propriétaire du compartiment afin qu'ils puissent effectuer des requêtes. Dans ce cas, utilisez une politique de compartiment pour accorder l'accès.

> **Note**
>
> Pour plus d'informations sur l'accès entre comptes aux catalogues de AWS Glue données d'Athena, consultez. Accès entre comptes aux catalogues de données AWS Glue

L'exemple suivant de politique de compartiment, créée et appliquée au compartiment `s3://DOC-EXAMPLE-BUCKET` par le propriétaire du compartiment, accorde l'accès à tous les utilisateurs du compte `123456789123`, qui est un compte différent.

```
{
   "Version": "2012-10-17",
   "Id": "MyPolicyID",
   "Statement": [
      {
          "Sid": "MyStatementSid",
```

```
            "Effect": "Allow",
            "Principal": {
               "AWS": "arn:aws:iam::123456789123:root"
            },
            "Action": [
               "s3:GetBucketLocation",
               "s3:GetObject",
               "s3:ListBucket",
               "s3:ListBucketMultipartUploads",
               "s3:ListMultipartUploadParts",
               "s3:AbortMultipartUpload"
            ],
            "Resource": [
               "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
               "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
         }
      ]
}
```

Pour accorder l'accès à un utilisateur particulier d'un compte, remplacez la clé `Principal` par une clé qui spécifie l'utilisateur à la place de `root`. Par exemple, pour le profil utilisateur `Dave`, utilisez `arn:aws:iam::123456789123:user/Dave`.

Accès entre comptes à un compartiment chiffré à l'aide d'une clé personnalisée AWS KMS

Si vous possédez un compartiment Amazon S3 chiffré à l'aide d'une clé custom AWS Key Management Service (AWS KMS), vous devrez peut-être autoriser l'accès à celui-ci aux utilisateurs d'un autre compte Amazon Web Services.

L'accès à un compartiment AWS KMS chiffré du compte A à un utilisateur du compte B nécessite les autorisations suivantes :

- La politique de compartiment du compte A doit accorder l'accès au rôle assumé par le compte B.
- La politique AWS KMS clé du compte A doit accorder l'accès au rôle assumé par l'utilisateur dans le compte B.
- Le rôle AWS Identity and Access Management (IAM) assumé par le compte B doit accorder l'accès au bucket et à la clé du compte A.

Les procédures suivantes décrivent comment accorder chacune de ces autorisations.

Pour accorder l'accès au compartiment dans le compte A à l'utilisateur dans le compte B

- Dans le compte A, consultez la politique de compartiment S3 et confirmez qu'il existe une instruction qui autorise l'accès à partir de l'ID de compte du compte B.

  Par exemple, la politique de compartiment suivante permet à `s3:GetObject` d'accéder à l'ID de compte `111122223333` :

```
{
  "Id": "ExamplePolicy1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExampleStmt1",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
      "Principal": {
        "AWS": [
          "111122223333"
        ]
      }
    }
  ]
}
```

Pour accorder l'accès à l'utilisateur du compte b à partir de la politique AWS KMS clé du compte a

1. Dans la politique AWS KMS clé du compte A, accordez au rôle assumé par le compte B des autorisations pour les actions suivantes :

   - `kms:Encrypt`
   - `kms:Decrypt`
   - `kms:ReEncrypt*`
   - `kms:GenerateDataKey*`
   - `kms:DescribeKey`

L'exemple suivant accorde l'accès à la clé à un seul rôle IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/role_name"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        }
    ]
}
```

2. À partir du compte A, passez en revue la politique clé à l'aide de la vue des AWS Management Console politiques.
3. Dans la politique de clé, vérifiez que l'instruction suivante indique le compte B en tant que principal.

```
"Sid": "Allow use of the key"
```

4. Si l'instruction `"Sid": "Allow use of the key"` n'est pas présente, effectuez les opérations suivantes :

   a. Basculez pour afficher la politique de clé à l'aide de la vue par défaut de la console.

   b. Ajoutez l'ID de compte du compte B en tant que compte externe avec accès à la clé.

Accorder l'accès au compartiment et à la clé du compte A à partir du rôle IAM assumé par le compte B

1. Depuis le compte B, ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Ouvrez le rôle IAM associé à l'utilisateur dans le compte B.
3. Consultez la liste des politiques d'autorisation appliquées au rôle IAM.
4. Assurez-vous qu'une politique est appliquée qui accorde l'accès au compartiment.

   L'exemple d'instruction suivant accorde au rôle IAM l'accès aux opérations `s3:GetObject` et `s3:PutObject` dans le compartiment `DOC-EXAMPLE-BUCKET` :

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "ExampleStmt2",
         "Action": [
           "s3:GetObject",
           "s3:PutObject"
         ],
         "Effect": "Allow",
         "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
       }
     ]
   }
   ```

5. Assurez-vous qu'une politique est appliquée qui accorde l'accès à la clé.

   > **Note**
   >
   > Si le rôle IAM assumé par le compte B dispose déjà d'un accès administrateur, il n'est pas nécessaire d'accorder l'accès à la clé à partir des politiques IAM de l'utilisateur.

   L'exemple d'instruction suivant accorde au rôle IAM l'accès à la clé `arn:aws:kms:us-west-2:123456789098:key/111aa2bb-333c-4d44-5555-a111bb2c33dd`.

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

```
    {
      "Sid": "ExampleStmt3",
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:Encrypt",
        "kms:GenerateDataKey",
        "kms:ReEncrypt*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:kms:us-west-2:123456789098:key/111aa2bb-333c-4d44-5555-
a111bb2c33dd"
    }
  ]
}
```

Accès inter-comptes aux objets de compartiment

Les objets qui sont téléchargés par un compte (compte C) autre que le compte propriétaire du compartiment (compte A) peuvent nécessiter des ACL explicites de niveau objet qui accordent un accès en lecture au compte d'interrogation (compte B). Pour éviter cette exigence, le compte C doit assumer un rôle dans le compte A avant de placer des objets dans le compartiment de ce dernier. Pour plus d'informations, consultez Comment puis-je fournir un accès inter-comptes aux objets stockés dans des compartiments Simple Storage Service (Amazon S3) ?.

Accès détaillé aux bases de données et aux tables du AWS Glue Data Catalog

Si vous l'utilisez AWS Glue Data Catalog avec Amazon Athena, vous pouvez définir des politiques au niveau des ressources pour les objets du catalogue de données de base de données et de tables utilisés dans Athena.

> **Note**
>
> Le terme « contrôle d'accès précis » est utilisé ici en référence à la sécurité au niveau des bases de données et des tableaux. Pour plus d'informations sur la sécurité au niveau des colonnes, des lignes et des cellules, consultez Filtrage des données et sécurité au niveau de la cellule dans Lake Formation.

Vous définissez les autorisations au niveau des ressources dans les politiques IAM basées sur l'identité.

> **Important**
>
> Cette section décrit les autorisations au niveau des ressources dans les politiques IAM basées sur l'identité. Celles-ci sont différentes des politiques basées sur les ressources. Pour plus d'informations sur les différences, consultez la rubrique Politiques basées sur l'identité et politiques basées sur les ressources du Guide de l'utilisateur IAM.

Pour plus d'informations sur ces tâches, consultez les rubriques suivantes :

Pour exécuter cette tâche	Consultez la rubrique suivante :
Créer une politique IAM qui définit un accès précis aux ressources	Création de politiques IAM dans le Guide de l'utilisateur IAM.
En savoir plus sur les politiques basées sur l'identité IAM utilisées dans AWS Glue	Politique basées sur l'identité (politiques IAM) dans le Guide du développeur AWS Glue .

Dans cette section :

- Limites
- AWS Glue accès à votre catalogue et à votre base de données par Région AWS
- Partitions de table et versions dans AWS Glue
- Exemples d'autorisations granulaires aux tables et bases de données

Limites

Tenez compte des limitations suivantes lorsque vous utilisez le contrôle d'accès précis avec le AWS Glue Data Catalog et Athena :

- Les groupes de travail Athena compatibles avec IAM Identity Center nécessitent que Lake Formation soit configuré pour utiliser les identités IAM Identity Center. Pour obtenir plus d'informations sur la configuration, consultez la rubrique Integrating IAM Identity Center dans le Guide du développeur AWS Lake Formation .
- Vous pouvez limiter l'accès uniquement aux bases de données et tables. Des contrôles précis des accès s'appliquent au niveau de la table et vous ne pouvez pas limiter l'accès aux partitions individuelles au sein d'une table. Pour plus d'informations, consultez Partitions de table et versions dans AWS Glue.
- AWS Glue Data Catalog Il contient les ressources suivantes : `CATALOGDATABASE`,`TABLE`, et`FUNCTION`.

> **Note**
>
> Dans cette liste, les ressources communes à Athéna et au AWS Glue Data Catalog sont `TABLEDATABASE`, et `CATALOG` pour chaque compte. `Functionest` spécifique à AWS Glue. Pour supprimer des actions dans Athena, vous devez inclure des autorisations dans les actions AWS Glue . veuillez consulter Exemples d'autorisations granulaires aux tables et bases de données.

La hiérarchie est la suivante : `CATALOG` est un ancêtre de toutes les `DATABASES` dans chaque compte, et chaque `DATABASE` est un ancêtre pour toutes ses `TABLES` et `FUNCTIONS`. Par exemple, pour une table nommée `table_test` qui appartient à une base de données db dans le catalogue de votre compte, ses ancêtres sont db et le catalogue dans votre compte. Pour la base de données db, son ancêtre est le catalogue dans votre compte, et ses descendants sont les tables et les fonctions. Pour plus d'informations sur la structure hiérarchique des ressources, consultez la section relative à la liste des ARN dans le catalogue de données dans le Guide du développeur AWS Glue .

- Pour toute action Athena de non-suppression sur une ressource, telle que `CREATE DATABASE`, `CREATE TABLE`, `SHOW DATABASE`, `SHOW TABLE` ou `ALTER TABLE`, vous devez disposer des autorisations nécessaires pour appeler cette action sur la ressource (table ou base de données) et tous les ancêtres de la ressource dans le catalogue de données. Par exemple, pour une table, ses ancêtres sont la base de données à laquelle elle appartient, et le catalogue du compte. Pour une base de données, son ancêtre est le catalogue du compte. veuillez consulter Exemples d'autorisations granulaires aux tables et bases de données.

- Pour une action de suppression dans Athena, telle que `DROP DATABASE` ou `DROP TABLE`, vous devez également disposer des autorisations nécessaires pour appeler l'action de suppression sur tous les ancêtres et descendants de la ressource dans le catalogue de données. Par exemple, pour supprimer une base de données, vous avez besoin des autorisations sur la base de données, du catalogue, qui est son ancêtre, et de toutes les tables et fonctions définies par l'utilisateur, qui sont ses descendants. Une table n'a pas de descendants. Pour que vous puissiez exécuter `DROP TABLE`, vous avez besoin des autorisations pour cette action sur la table, de la base de données à laquelle elle appartient et du catalogue. veuillez consulter Exemples d'autorisations granulaires aux tables et bases de données.

AWS Glue accès à votre catalogue et à votre base de données par Région AWS

Pour qu'Athéna puisse travailler avec le AWS Glue, une politique autorisant l'accès à votre base de données et AWS Glue Data Catalog à celle de votre compte Région AWS est requise. Pour créer des bases de données, l'autorisation `CreateDatabase` est également requise. Dans l'exemple de politique suivant, remplacez l' Région AWS Compte AWS ID et le nom de la base de données par les vôtres.

```
{
   "Sid": "DatabasePermissions",
   "Effect": "Allow",
   "Action": [
      "glue:GetDatabase",
      "glue:GetDatabases",
      "glue:CreateDatabase"
   ],
   "Resource": [
     "arn:aws:glue:us-east-1:123456789012:catalog",
     "arn:aws:glue:us-east-1:123456789012:database/default"
   ]
}
```

Partitions de table et versions dans AWS Glue

Dans AWS Glue, les tables peuvent avoir des partitions et des versions. Les versions de table et les partitions ne sont pas considérées comme des ressources indépendantes dans AWS Glue. L'accès aux versions et partitions de table est accordé en octroyant l'accès à la table et aux ressources ancêtres de la table.

Dans le cadre d'un contrôle d'accès détaillé, les autorisations d'accès suivantes s'appliquent :

- Les contrôles précis des accès s'appliquent au niveau de la table. Vous pouvez limiter l'accès uniquement aux bases de données et tables. Par exemple, si vous autorisez l'accès à une table partitionnée, cet accès s'applique à toutes les partitions de la table. Vous ne pouvez pas limiter l'accès aux partitions individuelles d'une table.

  > **Important**
  >
  > Pour exécuter des actions AWS Glue sur des partitions, des autorisations pour les actions de partition sont requises au niveau du catalogue, de la base de données et de la table. L'accès aux partitions d'une table n'est pas suffisant. Par exemple, pour exécuter `GetPartitions` sur une table `myTable` dans la base de données `myDB`, vous devez accorder des autorisations pour `glue:GetPartitions` au catalogue, à la base de données `myDB` et aux ressources `myTable`.

- Les contrôles précis des accès ne s'appliquent pas aux versions de table. Comme pour les partitions, l'accès aux versions précédentes d'une table est accordé via l'accès aux API de version de table AWS Glue présentes dans la table et aux ancêtres de la table.

Pour plus d'informations sur les autorisations relatives AWS Glue aux actions, voir Autorisations d'AWS Glue API : référence sur les actions et les ressources dans le guide du AWS Glue développeur.

Exemples d'autorisations granulaires aux tables et bases de données

Le tableau suivant répertorie des exemples de politiques IAM basées sur l'identité qui permettent un accès précis aux bases de données et tables dans Athena. Nous vous recommandons de commencer avec ces exemples et, en fonction de vos besoins, ajustez-les pour autoriser ou refuser des actions spécifiques à certaines bases de données et tables.

Ces exemples incluent l'accès aux bases de données et aux catalogues afin qu'Athéna AWS Glue et moi puissions travailler ensemble. Pour plusieurs AWS régions, incluez des politiques similaires pour chacune de vos bases de données et catalogues, une ligne pour chaque région.

En outre, remplacez la base de données `example_db` et la table `test` par les noms de votre base de données et de votre table.

Instruction DDL	Exemple d'une politique d'accès IAM accordant l'accès à la ressource
ALTER DATABASE	Permet de modifier les propriétés de la base de données `example_db` . `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:UpdateDatabase" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db " ] }`
CREATE DATABASE	Permet de créer la base de données nommée `example_db` . `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:CreateDatabase" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db " ] }`
CREATE TABLE	Permet de créer une table nommée `test` dans la base de données `example_db` . `{ "Sid": "DatabasePermissions", "Effect": "Allow", "Action": [ "glue:GetDatabase",`

Instruction DDL	Exemple d'une politique d'accès IAM accordant l'accès à la ressource

```
      "glue:GetDatabases"
   ],
   "Resource": [
     "arn:aws:glue: us-east-1 :123456789012 :catalog",
     "arn:aws:glue: us-east-1 :123456789012 :database
/ example_db "
   ]
},
{
   "Sid": "TablePermissions",
   "Effect": "Allow",
   "Action": [
      "glue:GetTables",
      "glue:GetTable",
      "glue:GetPartitions",
      "glue:CreateTable"
   ],
   "Resource": [
     "arn:aws:glue: us-east-1 :123456789012 :catalog",
     "arn:aws:glue: us-east-1 :123456789012 :database
/ example_db ",
     "arn:aws:glue: us-east-1 :123456789012 :table/example_d
b /test"
   ]
}
```

Instruction DDL	Exemple d'une politique d'accès IAM accordant l'accès à la ressource
DROP DATABASE	Permet de supprimer la base de données `example_db` , y compris toutes ses tables. `{ "Effect": "Allow", "Action": ["glue:GetDatabase", "glue:DeleteDatabase", "glue:GetTables", "glue:GetTable", "glue:DeleteTable"], "Resource": ["arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db ", "arn:aws:glue: us-east-1 :123456789012 :table/example_db /*", "arn:aws:glue: us-east-1 :123456789012 :userDefinedFunction/ example_db /*"] }`

Instruction DDL	Exemple d'une politique d'accès IAM accordant l'accès à la ressource
DROP TABLE	Permet de supprimer une table partitionnée nommée `test` dans la base de données `example_db` . Si votre table n'a pas de partitions, n'incluez pas d'actions de partition. `{ "Effect": "Allow", "Action": ["glue:GetDatabase", "glue:GetTable", "glue:DeleteTable", "glue:GetPartitions", "glue:GetPartition", "glue:DeletePartition"], "Resource": ["arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db ", "arn:aws:glue: us-east-1 :123456789012 :table/example_db /test"] }`

Instruction DDL	Exemple d'une politique d'accès IAM accordant l'accès à la ressource
MSCK REPAIR TABLE	Permet de mettre à jour les métadonnées du catalogue après avoir ajouté des partitions compatibles Hive à la table nommée `test` dans la base de données `example_db` . `{ "Effect": "Allow", "Action": ["glue:GetDatabase", "glue:CreateDatabase", "glue:GetTable", "glue:GetPartitions", "glue:GetPartition", "glue:BatchCreatePartition"], "Resource": ["arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db ", "arn:aws:glue: us-east-1 :123456789 012 :table/example_db /test"] }`
SHOW DATABASES	Permet de répertorier toutes les bases de données dans le AWS Glue Data Catalog. `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:GetDatabases" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database/*" ] }`

Instruction DDL	Exemple d'une politique d'accès IAM accordant l'accès à la ressource
SHOW TABLES	Permet de répertorier toutes les tables de la base de données `example_db` . `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:GetTables" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db ", "arn:aws:glue: us-east-1 :123456789012 :table/example_db /*" ] }`

Accès entre comptes aux catalogues de données AWS Glue

Vous pouvez utiliser la fonction de AWS Glue catalogue multi-comptes d'Athena pour enregistrer un AWS Glue catalogue à partir d'un compte autre que le vôtre. Après avoir configuré les autorisations IAM requises AWS Glue et enregistré le catalogue en tant que ressource DataCatalogAthena, vous pouvez utiliser Athena pour exécuter des requêtes entre comptes. Pour plus d'informations sur l'utilisation de la console Athena pour enregistrer un catalogue à partir d'un autre compte, consultez Enregistrer un compte AWS Glue Data Catalog depuis un autre compte.

Pour plus d'informations sur l'accès multicompte dans AWS Glue, consultez la section Octroi d'un accès multicompte dans le guide du AWS Glue développeur.

Avant de commencer

Étant donné que cette fonction utilise les API et les fonctionnalités existantes des ressources `DataCatalog` Athena pour permettre un accès inter-comptes, nous vous recommandons de lire les ressources suivantes avant de commencer :

- Connexion aux sources de données- Contient des rubriques sur l'utilisation d'Athena avec des sources de catalogue de données AWS Glue, Hive ou Lambda.

- Exemple de politiques de catalogue de données – Indique comment écrire des politiques qui contrôlent l'accès aux catalogues de données.
- Utilisation des AWS CLI métastores with Hive- Montre comment utiliser les métastores AWS CLI with Hive, mais contient des cas d'utilisation applicables à d'autres sources de données.

Considérations et restrictions

Actuellement, l'accès au AWS Glue catalogue entre comptes Athena présente les limites suivantes :

- Cette fonctionnalité n'est disponible que Régions AWS lorsque la version 2 ou ultérieure du moteur Athena est prise en charge. Pour plus d'informations sur les versions du moteur Athena, voir Gestion des versions du moteur Athena. Pour mettre à niveau la version du moteur pour un groupe de travail, veuillez consulter la rubrique Modification des versions du moteur Athena.
- Lorsque vous enregistrez un autre compte sur votre compte, vous créez une `DataCatalog` ressource régionale liée aux données de l'autre compte dans cette région en particulier uniquement. AWS Glue Data Catalog
- Actuellement, les instructions `CREATE VIEW` qui comprennent un catalogue AWS Glue inter-comptes ne sont pas prises en charge.
- Les catalogues chiffrés à l'aide de clés AWS gérées ne peuvent pas être consultés sur plusieurs comptes. Pour les catalogues que vous souhaitez interroger sur plusieurs comptes, utilisez plutôt des clés gérées par le client (`KMS_CMK`). Pour plus d'informations sur les différences entre les clés gérées par le client et les clés AWS gérées, consultez la section Clés et AWS clés client dans le guide du AWS Key Management Service développeur.

Premiers pas

Dans le scénario suivant, le compte « emprunteur » (666666666666) souhaite exécuter une `SELECT` requête faisant référence au AWS Glue catalogue appartenant au compte « propriétaire » (99999999999999), comme dans l'exemple suivant :

```
SELECT * FROM ownerCatalog.tpch1000.customer
```

Dans la procédure suivante, les étapes 1a et 1b montrent comment donner au compte emprunteur l'accès aux AWS Glue ressources du compte propriétaire, tant du côté de l'emprunteur que du côté du propriétaire. L'exemple accorde l'accès à la base de données `tpch1000` et à la table `customer`. Modifiez ces exemples de noms pour répondre à vos besoins.

Étape 1a : créer un rôle d'emprunteur avec une politique d'accès aux ressources du AWS Glue propriétaire

Pour créer un rôle de compte d'emprunteur avec une politique d'accès aux AWS Glue ressources du compte propriétaire, vous pouvez utiliser la console AWS Identity and Access Management (IAM) ou l'API IAM. La procédure suivante utilise la console IAM.

Pour créer un rôle d'emprunteur et une politique permettant d'accéder aux ressources du AWS Glue compte propriétaire

1. Connectez-vous à la console IAM à l'adresse https://console.aws.amazon.com/iam/ depuis le compte de l'emprunteur.
2. Dans le panneau de navigation, développez Gestion des accès, puis choisissez Politiques.
3. Choisissez Créer une politique.
4. Dans Éditeur de politique, choisissez JSON.
5. Dans l'éditeur de stratégie, entrez la politique suivante, puis modifiez-la en fonction de vos besoins :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "glue:*",
            "Resource": [
                "arn:aws:glue:us-east-1:999999999999:catalog",
                "arn:aws:glue:us-east-1:999999999999:database/tpch1000",
                "arn:aws:glue:us-east-1:999999999999:table/tpch1000/customer"
            ]
        }
    ]
}
```

6. Choisissez Suivant.
7. Sur la page Réviser et créer, dans Nom de la politique, entrez le nom de la stratégie (par exemple,**`CrossGluePolicyForBorrowerRole`**).
8. Choisissez Créer une politique.
9. Dans le panneau de navigation, choisissez Roles (Rôles).
10. Sélectionnez Create role (Créer un rôle).

11. Sur la page Sélectionner une entité de confiance, choisissez Compte AWS, puis cliquez sur Suivant.
12. Sur la page Ajouter des autorisations, entrez le nom de la politique que vous avez créée dans le champ de recherche (par exemple,**`CrossGluePolicyForBorrowerRole`**).
13. Cochez la case à côté du nom de la politique, puis choisissez Next.
14. Sur la page Name, review, and create (Nommer, réviser et créer) pour le Role name (nom de rôle), saisissez un nom de rôle (par exemple **`CrossGlueBorrowerRole`**).
15. Sélectionnez Créer un rôle.

Étape 1b : créer une politique du propriétaire pour accorder AWS Glue l'accès à l'emprunteur

Pour accorder l' AWS Glue accès depuis le compte propriétaire (999999999999) au rôle de l'emprunteur, vous pouvez utiliser la console ou l' AWS Glue opération API. AWS Glue PutResourcePolicy La procédure suivante utilise la AWS Glue console.

Pour autoriser le propriétaire à AWS Glue accéder au compte de l'emprunteur

1. Connectez-vous à la AWS Glue console à l'adresse https://console.aws.amazon.com/glue/ depuis le compte du propriétaire.
2. Dans le panneau de navigation, développez Catalogue de données, puis choisissez Paramètres du catalogue.
3. Dans Autorisations, saisissez une politique telle que la suivante. Pour *`rolename`*, entrez le rôle créé par l'emprunteur à l'étape 1a (par exemple,). **`CrossGlueBorrowerRole`** Si vous voulez augmenter la portée de l'autorisation, vous pouvez utiliser le caractère générique * pour les types de ressources base de données et table.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::666666666666:user/username",
                    "arn:aws:iam::666666666666:role/rolename"
                ]
            },
            "Action": "glue:*",
```

```
            "Resource": [
                "arn:aws:glue:us-east-1:999999999999:catalog",
                "arn:aws:glue:us-east-1:999999999999:database/tpch1000",
                "arn:aws:glue:us-east-1:999999999999:table/tpch1000/customer"
            ]
        }
    ]
}
```

Une fois que vous aurez terminé, nous vous recommandons d'utiliser l'AWS Glue API pour effectuer des tests d'appels entre comptes afin de confirmer que les autorisations sont configurées comme prévu.

Étape 2 : L'emprunteur enregistre le AWS Glue Data Catalog compte qui appartient au propriétaire

La procédure suivante vous montre comment utiliser la console Athena pour configurer le AWS Glue Data Catalog dans le compte Amazon Web Services propriétaire en tant que source de données. Pour obtenir des informations sur l'utilisation des opérations API au lieu de la console pour enregistrer le catalogue, consultez Utilisation de l'API pour enregistrer un catalogue de données Athena appartenant au compte propriétaire.

Pour enregistrer un compte AWS Glue Data Catalog appartenant à un autre compte

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Développez Administration, puis choisissez Sources de données.
4. Dans le coin supérieur droit, choisissez Create data source (Créer une source de données).
5. Sur la page Choisir une source de données, pour Sources de données, sélectionnez S3 - AWS Glue Data Catalog, puis Next.
6. Sur la page Enter data source details (Saisir les détails de la source de données), dans la section AWS Glue Data Catalog, pour Choisir un AWS Glue Data Catalog, choisissez AWS Glue Data Catalog dans un autre compte.
7. Pour Data source details (Détails de la source de données), saisissez les informations suivantes :

   - Data source name (Nom de la source de données) – Saisissez le nom que vous souhaitez utiliser dans vos requêtes SQL pour faire référence au catalogue de données dans l'autre compte.
   - Description – (Facultative) Saisissez une description du catalogue de données dans l'autre compte.
   - ID du catalogue – Saisissez l'ID de compte Amazon Web Services à 12 chiffres du compte auquel appartient le catalogue de données. L'ID de compte Amazon Web Services est l'ID de catalogue.

8. (Facultatif) Développez Identifications, saisissez les paires clé-valeur que vous voulez associer à la source de données. Pour en savoir plus sur les identifications, consultez Étiquetage des ressources Athena.
9. Choisissez Suivant.
10. Sur la page Review and create (Vérifier et créer), vérifiez les informations que vous avez fournies, puis choisissez Create data source (Créer une source de données). La page Data source details (Détails de la source de données) répertorie les bases de données et les balises du catalogue de données que vous avez enregistré.
11. Choisissez Sources de données. Le catalogue de données que vous avez enregistré est répertorié dans la colonne Data source name (Nom de la source de données).
12. Pour afficher ou modifier les informations relatives au catalogue de données, choisissez le catalogue, puis choisissez Actions, Edit (Modifier).
13. Pour supprimer le nouveau catalogue de données, choisissez le catalogue, puis choisissez Actions, Delete (Supprimer).

Étape 3 : l'emprunteur soumet une requête

L'emprunteur soumet une requête qui fait référence au catalogue à l'aide du *catalogue*. *base de données*. syntaxe de *table*, comme dans l'exemple suivant :

```
SELECT * FROM ownerCatalog.tpch1000.customer
```

Au lieu d'utiliser la syntaxe complète, l'emprunteur peut également spécifier le catalogue de manière contextuelle en le transmettant via le. QueryExecutionContext

Autorisations supplémentaires Simple Storage Service (Amazon S3)

- Si le compte emprunteur utilise une requête Athena pour écrire de nouvelles données dans une table du compte propriétaire, le propriétaire n'aura pas automatiquement accès à ces données dans Amazon S3, même si la table existe dans le compte du propriétaire. Cela est dû au fait que l'emprunteur est le propriétaire de l'objet des informations dans Amazon S3, sauf configuration contraire. Pour accorder au propriétaire l'accès aux données, définissez les autorisations sur les objets en conséquence dans le cadre d'une étape supplémentaire.
- Certaines opérations DDL inter-comptes comme MSCK REPAIR TABLE nécessitent des autorisations Simple Storage Service (Amazon S3). Par exemple, si le compte emprunteur effectue une `MSCK REPAIR` opération entre comptes sur une table du compte propriétaire dont les données se trouvent dans un compartiment S3 du compte propriétaire, ce compartiment doit accorder des autorisations au rôle assumé par l'emprunteur pour que la requête aboutisse.

Pour plus d'informations sur l'octroi des autorisations de compartiment, consultez la section Comment je définir les autorisations de compartiment ACL ? du Guide de l'utilisateur Amazon Simple Storage Service.

Utilisation dynamique d'un catalogue

Dans certains cas, vous souhaitez peut-être effectuer rapidement des tests sur un catalogue AWS Glue inter-comptes sans passer par l'étape préalable de l'enregistrement. Vous pouvez effectuer dynamiquement des requêtes inter-comptes sans créer l'objet ressource `DataCatalog` si les autorisations IAM et Simple Storage Service (Amazon S3) requises sont correctement configurées comme décrit précédemment dans ce document.

Pour référencer explicitement un catalogue sans enregistrement, utilisez la syntaxe de l'exemple suivant :

```
SELECT * FROM "glue:arn:aws:glue:us-east-1:999999999999:catalog".tpch1000.customer
```

Utilisez le format « `glue:<arn>` », où *<arn>* est l'ARN AWS Glue Data Catalog que vous voulez utiliser. Dans l'exemple, Athena utilise cette syntaxe pour pointer dynamiquement vers le catalogue de AWS Glue données du compte 999999999999, comme si vous aviez créé un objet distinct pour celui-ci. `DataCatalog`

Remarques sur l'utilisation des catalogues dynamiques

Lorsque vous utilisez des catalogues dynamiques, rappelez-vous les points suivants.

- L'utilisation d'un catalogue dynamique nécessite les autorisations IAM que vous utilisez normalement pour les opérations de l'API Athena Data Catalog (Catalogue de données Athena). La principale différence est que le nom de la ressource Data Catalog (Catalogue de données) suit la convention d'appellation `glue:*`.
- L'ARN du catalogue doit appartenir à la même région que celle où la requête est exécutée.
- Lorsqu'un catalogue dynamique est utilisé dans une requête ou une vue DML, il doit être entouré de guillemets doubles échappés (\"). Lorsqu'un catalogue dynamique est utilisé dans une requête DDL, il doit être entouré de guillemets simples inversés (`).

Utilisation de l'API pour enregistrer un catalogue de données Athena appartenant au compte propriétaire

Au lieu d'utiliser la console Athena comme décrit à l'étape 2, il est possible d'utiliser les opérations API pour enregistrer le catalogue de données appartenant au compte propriétaire.

Le créateur de la DataCatalogressource Athena doit disposer des autorisations nécessaires pour exécuter l'opération d'API CreateDataCatalogAthena. En fonction de vos besoins, l'accès à des opérations API supplémentaires peut être nécessaire. Pour plus d'informations, consultez Exemple de politiques de catalogue de données.

Le corps de `CreateDataCatalog` demande suivant enregistre un AWS Glue catalogue pour un accès entre comptes :

```
# Example CreateDataCatalog request to register a cross-account Glue catalog:
{
    "Description": "Cross-account Glue catalog",
    "Name": "ownerCatalog",
    "Parameters": {"catalog-id" : "999999999999"  # Owner's account ID
```

```
    },
    "Type": "GLUE"
}
```

L'exemple de code suivant utilise un client Java pour créer l'objet `DataCatalog`.

```
# Sample code to create the DataCatalog through Java client
CreateDataCatalogRequest request = new CreateDataCatalogRequest()
    .withName("ownerCatalog")
    .withType(DataCatalogType.GLUE)
    .withParameters(ImmutableMap.of("catalog-id", "999999999999"));

athenaClient.createDataCatalog(request);
```

Après ces étapes, l'emprunteur devrait voir `ownerCatalog` quand il appelle l'opération ListDataCatalogsAPI.

Ressources supplémentaires

- Enregistrer un compte AWS Glue Data Catalog depuis un autre compte
- Configurez l'accès entre comptes à un compte partagé à AWS Glue Data Catalog l'aide d'Amazon Athena dans AWS le guide Prescriptive Guidance Patterns.
- Interrogez plusieurs comptes AWS Glue Data Catalogà l'aide d'Amazon Athena sur AWS le blog Big Data
- Octroi d'un accès intercompte dans le Guide du développeur AWS Glue

Accès depuis Athena aux métadonnées cryptées dans le AWS Glue Data Catalog

Si vous l'utilisez AWS Glue Data Catalog avec Amazon Athena, vous pouvez activer le chiffrement à l' AWS Glue Data Catalog aide de la AWS Glue console ou de l'API. Pour obtenir des informations, consultez la rubrique Chiffrement du catalogue de données dans le Guide du développeur AWS Glue .

S'il AWS Glue Data Catalog est crypté, vous devez ajouter les actions suivantes à toutes les politiques utilisées pour accéder à Athena :

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

```
 "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt",
           "kms:Encrypt"
      ],
     "Resource": "(arn of the key used to encrypt the catalog)"
   }
}
```

Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Accès aux groupes de travail et identifications

Un groupe de travail est une ressource gérée par Athena. Ainsi, si la politique de votre groupe de travail utilise des actions prenant `workgroup` comme entrée, vous devez préciser l'ARN du groupe de travail comme suit, où *`workgroup-name`* est le nom de votre groupe de travail :

```
"Resource": [arn:aws:athena:region:AWSAcctID:workgroup/workgroup-name]
```

Par exemple, pour un groupe de travail nommé `test_workgroup` dans la région `us-west-2` pour le compte Amazon Web Services `123456789012`, spécifiez le groupe de travail sous la forme d'une ressource à l'aide de l'ARN suivant :

```
"Resource":["arn:aws:athena:us-east-2:123456789012:workgroup/test_workgroup"]
```

Pour accéder aux groupes de travail compatibles avec la propagation sécurisée des identités (TIP), les utilisateurs d'IAM Identity Center doivent être affectés à `IdentityCenterApplicationArn` celui renvoyé par la réponse de l'action de l'API GetWorkGroupAthena.

- Pour une liste des politiques de groupes de travail, consultez the section called “Exemples de politiques de groupe de travail”.
- Pour obtenir une liste des politiques basées sur les identifications pour les groupes de travail, consultez Politiques de contrôle d'accès IAM basées sur les étiquettes.
- Pour plus d'informations sur la création de politiques IAM pour les groupes de travail, voir Politiques IAM pour l'accès aux groupes de travail.
- Pour obtenir une liste complète d'actions Amazon Athena, consultez les noms d'action d'API dans la Référence d'API Amazon Athena.

- Pour plus d'informations sur les politiques IAM, consultez la rubrique Création de politiques avec l'éditeur visuel du Guide de l'utilisateur IAM.

Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Autorisation d'accès aux instructions préparées

Cette rubrique traite des autorisations IAM pour les instructions préparées dans Amazon Athena. Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Pour plus d'informations sur les instructions préparées, voir Utilisation de requêtes paramétrées.

Les autorisations IAM suivantes sont requises pour créer, gérer et exécuter des instructions préparées.

```
athena:CreatePreparedStatement
athena:UpdatePreparedStatement
athena:GetPreparedStatement
athena:ListPreparedStatements
athena:DeletePreparedStatement
```

Utilisez ces autorisations comme indiqué dans le tableau suivant.

Pour	Utiliser ces autorisations
Exécuter une requête PREPARE	`athena:StartQueryExecution` `athena:CreatePreparedStatement`
Ré-exécuter une requête PREPARE pour mettre à jour une instruction préparée existante	`athena:StartQueryExecution` `athena:UpdatePreparedStatement`
Exécuter une requête EXECUTE	`athena:StartQueryExecution` `athena:GetPreparedStatement`
Exécuter une requête DEALLOCATE PREPARE	`athena:StartQueryExecution` `athena:DeletePreparedStatement`

Exemple

L'exemple de politique IAM suivant accorde des autorisations pour gérer et exécuter des instructions préparées sur un ID de compte et un groupe de travail spécifiés.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:StartQueryExecution",
                "athena:CreatePreparedStatement",
                "athena:UpdatePreparedStatement",
                "athena:GetPreparedStatement",
                "athena:DeletePreparedStatement",
                "athena:ListPreparedStatements"
            ],
            "Resource": [
                "arn:aws:athena:*:111122223333:workgroup/<workgroup-name>"
            ]
        }
    ]
}
```

Utilisation d'Athéna avec CalledVia des touches contextuelles

Lorsqu'un principal fait une demande à AWS, AWS rassemble les informations de la demande dans un contexte de demande qui évalue et autorise la demande. Vous pouvez utiliser l'élément `Condition` d'une politique JSON pour comparer des clés dans le contexte de demande avec les valeurs de clé spécifiées dans votre politique. Les clés de condition générales sont des clés de condition avec un préfixe `aws:`.

Clés contextuelle aws:CalledVia

Vous pouvez utiliser la clé contextuelle de condition générale aws:CalledVia pour comparer les services de la politique avec les services qui ont fait des demandes au nom du principal IAM (utilisateur ou rôle). Lorsqu'un principal adresse une demande à un Service AWS, ce service peut utiliser les informations d'identification du principal pour faire des demandes ultérieures à d'autres services. La clé `aws:CalledVia` contient une liste ordonnée des services de la chaîne ayant effectué des demandes pour le compte du principal.

En spécifiant un nom principal de service pour la clé de `aws:CalledVia` contexte, vous pouvez la rendre Service AWS spécifique. Par exemple, vous pouvez utiliser la clé de condition `aws:CalledVia` pour limiter les demandes à celles effectuées à partir d'Athena. Pour utiliser la clé de condition `aws:CalledVia` dans une politique avec Athena, vous devez spécifier le `athena.amazonaws.com` du nom du principal du service Athena, comme dans l'exemple suivant.

```
 ...
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:CalledVia": "athena.amazonaws.com"
        }
    }
...
```

Vous pouvez utiliser la clé contextuelle `aws:CalledVia` pour vous assurer que les appelants n'ont accès à une ressource (comme une fonction Lambda) que s'ils appellent la ressource à partir d'Athena.

> **Note**
>
> La clé de contexte `aws:CalledVia` n’est pas compatible avec la fonctionnalité de propagation d’identité approuvée.

Ajoutez une clé de CalledVia contexte facultative pour un accès détaillé à une fonction Lambda

Athena exige que l'appelant dispose d'autorisations `lambda:InvokeFunction` afin d'invoquer la fonction Lambda associée à la requête. L'instruction suivante permet un accès précis à une fonction Lambda afin que l'utilisateur puisse utiliser uniquement Athena pour invoquer la fonction Lambda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:OneAthenaLambdaFunction",
            "Condition": {
```

```
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "athena.amazonaws.com"
                }
            }
        }
    ]
}
```

L'exemple suivant montre l'ajout de l'instruction précédente à une politique qui permet à un utilisateur d'exécuter et de lire une requête fédérée. Les principals autorisés à effectuer ces actions peuvent exécuter des requêtes qui spécifient les catalogues Athena associés à une source de données fédérée. Cependant, le principal ne peut pas accéder à la fonction Lambda associée, à moins que la fonction ne soit invoquée par Athena.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "athena:GetWorkGroup",
                "s3:PutObject",
                "s3:GetObject",
                "athena:StartQueryExecution",
                "s3:AbortMultipartUpload",
                "athena:StopQueryExecution",
                "athena:GetQueryExecution",
                "athena:GetQueryResults",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:athena:*:111122223333:workgroup/WorkGroupName",
                "arn:aws:s3:::MyQueryResultsBucket/*",
                "arn:aws:s3:::MyLambdaSpillBucket/MyLambdaSpillPrefix*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "athena:ListWorkGroups",
            "Resource": "*"
        },
```

```
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action":
                [
                "s3:ListBucket",
                "s3:GetBucketLocation"
                ],
            "Resource": "arn:aws:s3:::MyLambdaSpillBucket"
        },
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": [
                "arn:aws:lambda:*:111122223333:function:OneAthenaLambdaFunction",
                "arn:aws:lambda:*:111122223333:function:AnotherAthenaLambdaFunction"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "athena.amazonaws.com"
                }
            }
        }
    ]
}
```

Pour plus d'informations sur les clés de condition `CalledVia`, consultez la rubrique Clés contextuelles de condition générale AWS du Guide de l'utilisateur IAM.

Autorisation d'accès à un connecteur de données Athena pour un métastore Hive externe

Les exemples de politique d'autorisation présentés dans cette rubrique illustrent les actions autorisées requises et les ressources pour lesquelles elles sont autorisées. Examinez attentivement ces politiques et modifiez-les en fonction de vos besoins avant d'attacher des politiques d'autorisation similaires aux identités IAM.

- Example Policy to Allow an IAM Principal to Query Data Using Athena Data Connector for External Hive Metastore
- Example Policy to Allow an IAM Principal to Create an Athena Data Connector for External Hive Metastore

Example – Autoriser un principal IAM à interroger des données à l'aide du connecteur de données Athena pour le métastore Hive externe

La politique suivante est attachée aux principals IAM en plus de AWS politique gérée : AmazonAthenaFullAccess, qui accorde un accès complet aux actions Athena.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "lambda:GetFunction",
                "lambda:GetLayerVersion",
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:111122223333:function:MyAthenaLambdaFunction",
                "arn:aws:lambda:*:111122223333:function:AnotherAthenaLambdaFunction",
                "arn:aws:lambda:*:111122223333:layer:MyAthenaLambdaLayer:*"
            ]
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": "arn:aws:s3:::MyLambdaSpillBucket/MyLambdaSpillLocation"
        }
    ]
}
```

Explication des autorisations

Actions autorisées	Explication
`"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:PutObject", "s3:ListMultipartUploadParts", "s3:AbortMultipartUpload"`	s3les actions permettent de lire et d'écrire dans la ressource spécifiée`"arn:aws:s3::: MyLambdaSpillBucket /MyLambdaSpillLocation "`, où *MyLambdaSpillLocation* identifie le bucket de déversement spécifié dans la configuration de la ou des fonctions Lambda invoquées. L'identifiant de *ressource arn:aws:lambda : *: my:Layer : MyAthenaLambdaLayer :* n'est requis que si vous utilisez une couche AWSAcctId* Lambda pour créer des dépendances d'exécution personnalisées afin de réduire la taille des artefacts fonctionnels au moment du déploiement. Le caractère * en dernière position est un caractère générique pour la version de la couche.
`"lambda:GetFunction", "lambda:GetLayerVersion", "lambda:InvokeFunction"`	Permet aux requêtes d'invoquer les AWS Lambda fonctions spécifiées dans le `Resource` bloc. Par exemple`arn:aws:lambda:*: MyAWSAcctId :function : MyAthenaLambdaFunction` , où *MyAthenaLambdaFunction* indique le nom d'une fonction Lambda à invoquer. Plusieurs fonctions peuvent être spécifiées comme indiqué dans l'exemple.

Example – Autoriser un principal IAM à créer un connecteur de données Athena pour le métastore Hive externe

La politique suivante est attachée aux principals IAM en plus de AWS politique gérée : AmazonAthenaFullAccess, qui accorde un accès complet aux actions Athena.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "lambda:GetFunction",
                "lambda:ListFunctions",
                "lambda:GetLayerVersion",
                "lambda:InvokeFunction",
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:PublishLayerVersion",
                "lambda:DeleteLayerVersion",
                "lambda:UpdateFunctionConfiguration",
                "lambda:PutFunctionConcurrency",
                "lambda:DeleteFunctionConcurrency"
            ],
            "Resource": "arn:aws:lambda:*:111122223333:
 function: MyAthenaLambdaFunctionsPrefix*"
        }
    ]
}
```

Explication des autorisations

Permet aux requêtes d'invoquer les AWS Lambda fonctions pour les AWS Lambda fonctions spécifiées dans le `Resource` bloc. Par exemple`arn:aws:lambda:*:`*`MyAWSAcctId`*`:function:`*`MyAthenaLambdaFunction`*, où *`MyAthenaLambdaFunction`*indique le nom d'une fonction Lambda à invoquer. Plusieurs fonctions peuvent être spécifiées comme indiqué dans l'exemple.

Autorisation d'accès des fonctions Lambda aux métastores Hive externes

Pour invoquer une fonction Lambda dans votre compte, vous devez créer un rôle disposant des autorisations suivantes :

- `AWSLambdaVPCAccessExecutionRole` – Autorisation rôle d'exécution AWS Lambda permettant de gérer des interfaces réseau élastiques qui connectent votre fonction à un VPC. Assurez-vous que vous disposez d'un nombre suffisant d'interfaces réseau et d'adresses IP disponibles.

- `AmazonAthenaFullAccess`— La politique AmazonAthenaFullAccessgérée accorde un accès complet à Athéna.
- Une politique Simple Storage Service (Amazon S3) pour permettre à la fonction Lambda d'écrire sur S3 et à Athena de lire à partir de S3.

Par exemple, la politique suivante définit l'autorisation pour l'emplacement de déversement `s3:\\mybucket\spill`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/spill"
            ]
        }
    ]
}
```

Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Création de fonctions Lambda

Pour créer une fonction Lambda dans votre compte, des autorisations de développement de fonction ou le rôle `AWSLambdaFullAccess` sont nécessaires. Pour de plus amples informations, veuillez consulter la rubrique Politiques IAM basées sur l'identité pour AWS Lambda.

Comme Athena utilise le AWS Serverless Application Repository pour créer des fonctions Lambda, le superutilisateur ou l'administrateur qui crée des fonctions Lambda doit également disposer de politiques IAM autorisant les requêtes fédérées Athena.

Opérations d'API d'enregistrement de catalogue et de métadonnées

Pour accéder à l'API d'enregistrement du catalogue et aux opérations de l'API de métadonnées, utilisez la politique AmazonAthenaFullAccess gérée. Si vous n'utilisez pas cette politique, ajoutez les opérations API suivantes à vos politiques Athena :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListDataCatalogs",
                "athena:GetDataCatalog",
                "athena:CreateDataCatalog",
                "athena:UpdateDataCatalog",
                "athena:DeleteDataCatalog",
                "athena:GetDatabase",
                "athena:ListDatabases",
                "athena:GetTableMetadata",
                "athena:ListTableMetadata"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Appel Lambda interrégional

Pour invoquer une fonction Lambda dans une région autre que celle dans laquelle vous exécutez les requêtes Athena, utilisez l'ARN complet de la fonction Lambda. Par défaut, Athena invoque des fonctions Lambda définies dans la même région. Si vous devez invoquer une fonction Lambda pour accéder à un métastore Hive dans une région autre que celle dans laquelle vous exécutez des requêtes Athena, vous devez fournir l'ARN complet de la fonction Lambda.

Par exemple, supposons que vous définissiez le catalogue `ehms` sur la région Europe (Francfort) `eu-central-1` pour utiliser la fonction Lambda suivante dans la région USA Est (Virginie du Nord).

```
arn:aws:lambda:us-east-1:111122223333:function:external-hms-service-new
```

Lorsque vous spécifiez l'ARN complet de cette manière, Athena peut appeler la fonction Lambda `external-hms-service-new` sur `us-east-1` pour récupérer les données du métastore Hive à partir de `eu-central-1`.

> **Note**
>
> Le catalogue `ehms` doit être enregistré dans la même région que celle où vous exécutez des requêtes Athena.

Appel de Lambda inter-comptes

Parfois, vous pouvez avoir besoin d'accéder à un métastore Hive à partir d'un autre compte. Par exemple, pour exécuter un métastore Hive, vous pouvez lancer un cluster EMR à partir d'un compte différent de celui que vous utilisez pour les requêtes Athena. Différents groupes ou équipes peuvent exécuter le métastore Hive avec différents comptes à l'intérieur de leur VPC. Vous pouvez également accéder aux métadonnées de différents métastores Hive de différents groupes ou équipes.

Athena utilise la prise en charge AWS Lambda de l'accès multi-comptes pour permettre l'accès multi-comptes aux métastores Hive.

> **Note**
>
> Notez que l'accès entre comptes pour Athena implique normalement un accès inter-comptes pour les métadonnées et les données dans Simple Storage Service (Amazon S3).

Imaginez le scénario suivant :

- Le compte `111122223333` configure la fonction Lambda `external-hms-service-new` sur us-east-1 dans Athena pour accéder à un métastore Hive exécuté sur un cluster EMR.
- Le compte `111122223333` veut autoriser le compte 444455556666 à accéder aux données du métastore Hive.

Pour autoriser `444455556666` un compte à accéder à la fonction Lambda`external-hms-service-new`, account `111122223333` utilise la commande suivante AWS CLI `add-permission`. La commande a été formatée pour être lisible.

```
$ aws --profile perf-test lambda add-permission
```

```
      --function-name external-hms-service-new
      --region us-east-1
      --statement-id Id-ehms-invocation2
      --action "lambda:InvokeFunction"
      --principal arn:aws:iam::444455556666:user/perf1-test
{
    "Statement": "{\"Sid\":\"Id-ehms-invocation2\",
                  \"Effect\":\"Allow\",
                  \"Principal\":{\"AWS\":\"arn:aws:iam::444455556666:user/perf1-test
\"},
                  \"Action\":\"lambda:InvokeFunction\",
                  \"Resource\":\"arn:aws:lambda:us-
east-1:111122223333:function:external-hms-service-new\"}"
}
```

Pour vérifier l'autorisation Lambda, utilisez la commande `get-policy`, comme dans l'exemple suivant. La commande a été formatée pour être lisible.

```
$ aws --profile perf-test lambda get-policy
      --function-name arn:aws:lambda:us-east-1:111122223333:function:external-hms-
service-new
      --region us-east-1
{
    "RevisionId": "711e93ea-9851-44c8-a09f-5f2a2829d40f",
    "Policy": "{\"Version\":\"2012-10-17\",
                \"Id\":\"default\",
                \"Statement\":[{\"Sid\":\"Id-ehms-invocation2\",
                               \"Effect\":\"Allow\",
                               \"Principal\":{\"AWS\":
\"arn:aws:iam::444455556666:user/perf1-test\"},
                               \"Action\":\"lambda:InvokeFunction\",
                               \"Resource\":\"arn:aws:lambda:us-
east-1:111122223333:function:external-hms-service-new\"}]}"
}
```

Après avoir ajouté l'autorisation, vous pouvez utiliser un ARN complet de la fonction Lambda sur `us-east-1` comme suit lorsque vous définissez le catalogue `ehms` :

```
arn:aws:lambda:us-east-1:111122223333:function:external-hms-service-new
```

Pour de plus amples informations sur l'appel entre régions, veuillez consulter Appel Lambda interrégional plus haut dans cette rubrique.

Octroi d'un accès entre comptes aux données

Avant de pouvoir exécuter des requêtes Athena, vous devez accorder l'accès inter-comptes aux données dans Simple Storage Service (Amazon S3). Vous pouvez effectuer cette opération de différentes manières :

- Mettez à jour la politique de la liste de contrôle d'accès du compartiment Simple Storage Service (Amazon S3) avec un ID utilisateur canonique.
- Ajoutez l'accès inter-comptes à la politique de compartiment Simple Storage Service (Amazon S3).

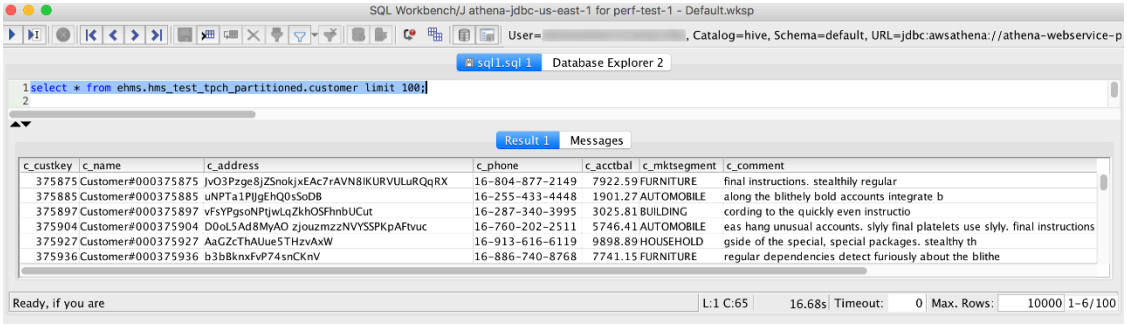
Par exemple, ajoutez la politique suivante à la politique de compartiment Simple Storage Service (Amazon S3) du compte `111122223333` pour permettre au compte `444455556666` de lire des données à partir de l'emplacement Simple Storage Service (Amazon S3) spécifié.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1234567890123",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:user/perf1-test"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::athena-test/lambda/dataset/*"
        }
    ]
}
```

> **Note**
>
> Il se peut que vous deviez accorder un accès inter-comptes à Simple Storage Service (Amazon S3) non seulement à vos données, mais aussi à votre emplacement de déversement Simple Storage Service (Amazon S3). Votre fonction Lambda déverse des données supplémentaires à l'emplacement de déversement lorsque la taille de l'objet de réponse dépasse un seuil donné. Veuillez consulter le début de cette rubrique pour obtenir un exemple de politique.

Dans l'exemple actuel, une fois que l'accès entre comptes est accordé à `444455556666`, `444455556666` peut utiliser le catalogue `ehms` dans son propre `account` pour interroger les tables qui sont définies dans le compte `111122223333`.

Dans l'exemple suivant, le profil SQL Workbench `perf-test-1` est pour le compte `444455556666`. La requête utilise le catalogue `ehms` pour accéder au métastore Hive et aux données Simple Storage Service (Amazon S3) du compte `111122223333`.



Exemple de politiques d'autorisation IAM pour autoriser la requête fédérée Athena

Les exemples de politique d'autorisation présentés dans cette rubrique illustrent les actions autorisées requises et les ressources pour lesquelles elles sont autorisées. Examinez attentivement ces politiques et modifiez-les en fonction de vos besoins avant de les attacher aux identités IAM.

Pour plus d'informations sur l'ajout des politiques aux identités IAM, consultez la rubrique Ajout et suppression des autorisations d'identité IAM du Guide de l'utilisateur IAM.

- Example policy to allow an IAM principal to run and return results using Athena Federated Query
- Example Policy to Allow an IAM Principal to Create a Data Source Connector

Example – Autoriser un principal IAM à exécuter et à renvoyer des résultats à l'aide de la requête fédérée Athena

La politique d'autorisations basée sur l'identité suivante autorise les actions dont un utilisateur ou un autre principal IAM a besoin pour utiliser la requête fédérée Athena. Les principals autorisés à effectuer ces actions peuvent exécuter des requêtes spécifiant les catalogues Athena associés à une source de données fédérée.

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

```
            "Sid": "Athena",
            "Effect": "Allow",
            "Action": [
                "athena:GetDataCatalog",
                "athena:GetQueryExecution",
                "athena:GetQueryResults",
                "athena:GetWorkGroup",
                "athena:StartQueryExecution",
                "athena:StopQueryExecution"
            ],
            "Resource": [
                "arn:aws:athena:*:111122223333:workgroup/WorkgroupName",
                "arn:aws:athena:aws_region:111122223333:datacatalog/DataCatalogName"
            ]
        },
        {
            "Sid": "ListAthenaWorkGroups",
            "Effect": "Allow",
            "Action": "athena:ListWorkGroups",
            "Resource": "*"
        },
        {
            "Sid": "Lambda",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": [
                "arn:aws:lambda:*:111122223333:function:OneAthenaLambdaFunction",
                "arn:aws:lambda:*:111122223333:function:AnotherAthenaLambdaFunction"
            ]
        },
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListMultipartUploadParts",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::MyLambdaSpillBucket",
                "arn:aws:s3:::MyLambdaSpillBucket/*",
```

```
                "arn:aws:s3:::MyQueryResultsBucket",
                "arn:aws:s3:::MyQueryResultsBucket/*"
            ]
        }
    ]
}
```

Explication des autorisations

Actions autorisées	Explication
`"athena:GetQueryExecution", "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StartQueryExecution", "athena:StopQueryExecution"`	Autorisations Athena nécessaires pour exécuter des requêtes fédérées.
`"athena:GetDataCatalog", "athena:GetQueryExecution," "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StartQueryExecution", "athena:StopQueryExecution"`	Autorisations Athena nécessaires pour exécuter des requêtes de vues fédérées. L'`GetDataCatalog` action est requise pour les vues.
`"lambda:InvokeFunction"`	Permet aux requêtes d'invoquer les AWS Lambda fonctions pour les AWS Lambda fonctions spécifiées dans le `Resource` bloc. Par exemple`arn:aws:lambda:*: MyAWSAcctId :function: MyAthenaLambdaFunction` , où `MyAthenaLambdaFunction` indique le nom d'une fonction Lambda à invoquer. Comme le montre l'exemple, plusieurs fonctions peuvent être spécifiées.
`"s3:AbortMultipartUpload", "s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:ListMultipartUploadParts",`	Les `s3:GetBucketLocation` autorisations `s3:ListBucket` et sont requises pour accéder au compartiment de sortie des requêtes pour les principaux IAM qui s'exécutent. `StartQueryExecution`

Actions autorisées	Explication
`"s3:PutObject"`	`s3:PutObject` `s3:ListMultipartUploadParts` , et `s3:AbortMultipartUpload` autorisez l'écriture des résultats de la requête dans tous les sous-dossiers du bucket de résultats de requête tel que spécifié par l'identifiant de `arn:aws:s3::: MyQueryResultsBucket /*` ressource, où se *MyQueryResultsBucket* trouve le bucket de résultats de requête Athena. Pour plus d'informations, consultez Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie. `s3:GetObject` permet de lire les résultats de la requête et l'historique des requêtes pour la ressource spécifiée sous la forme`arn:aws:s3::: MyQueryResultsBucket` , où se *MyQueryResultsBucket* trouve le bucket de résultats de requête Athena. `s3:GetObject` permet également de lire à partir de la ressource spécifiée sous la forme`"arn:aws:s3::: MyLambdaSpillBucket /MyLambdaSpillPrefix *"`, où *MyLambdaSpillPrefix* est spécifiée dans la configuration de la ou des fonctions Lambda invoquées.

Example – Autoriser un principal IAM à créer un connecteur de source de données

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
```

```
            "Action": [
                "lambda:CreateFunction",
                "lambda:ListVersionsByFunction",
                "iam:CreateRole",
                "lambda:GetFunctionConfiguration",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "lambda:PutFunctionConcurrency",
                "iam:PassRole",
                "iam:DetachRolePolicy",
                "lambda:ListTags",
                "iam:ListAttachedRolePolicies",
                "iam:DeleteRolePolicy",
                "lambda:DeleteFunction",
                "lambda:GetAlias",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetPolicy",
                "lambda:InvokeFunction",
                "lambda:GetFunction",
                "lambda:ListAliases",
                "lambda:UpdateFunctionConfiguration",
                "iam:DeleteRole",
                "lambda:UpdateFunctionCode",
                "s3:GetObject",
                "lambda:AddPermission",
                "iam:UpdateRole",
                "lambda:DeleteFunctionConcurrency",
                "lambda:RemovePermission",
                "iam:GetRolePolicy",
                "lambda:GetPolicy"
            ],
            "Resource": [

 "arn:aws:lambda:*:111122223333:function:MyAthenaLambdaFunctionsPrefix*",
                "arn:aws:s3:::awsserverlessrepo-changesets-1iiv3xa62ln3m/*",
                "arn:aws:iam::*:role/RoleName",
                "arn:aws:iam::111122223333:policy/*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
```

```
                "cloudformation:CreateUploadBucket",
                "cloudformation:DescribeStackDriftDetectionStatus",
                "cloudformation:ListExports",
                "cloudformation:ListStacks",
                "cloudformation:ListImports",
                "lambda:ListFunctions",
                "iam:ListRoles",
                "lambda:GetAccountSettings",
                "ec2:DescribeSecurityGroups",
                "cloudformation:EstimateTemplateCost",
                "ec2:DescribeVpcs",
                "lambda:ListEventSourceMappings",
                "cloudformation:DescribeAccountLimits",
                "ec2:DescribeSubnets",
                "cloudformation:CreateStackSet",
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "cloudformation:*",
            "Resource": [
                "arn:aws:cloudformation:*:111122223333:stack/aws-serverless-
repository-MyCFStackPrefix*/*",
                "arn:aws:cloudformation:*:111122223333:stack/
serverlessrepo-MyCFStackPrefix*/*",
                "arn:aws:cloudformation:*:*:transform/Serverless-*",
                "arn:aws:cloudformation:*:111122223333:stackset/aws-serverless-
repository-MyCFStackPrefix*:*",
                "arn:aws:cloudformation:*:111122223333:stackset/
serverlessrepo-MyCFStackPrefix*:*"
            ]
        },
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "serverlessrepo:*",
            "Resource": "arn:aws:serverlessrepo:*:*:applications/*"
        }
    ]
}
```

Explication des autorisations

Actions autorisées	Explication
`"lambda:CreateFunction", "lambda:ListVersionsByFunction", "lambda:GetFunctionConfiguration", "lambda:PutFunctionConcurrency", "lambda:ListTags", "lambda:DeleteFunction", "lambda:GetAlias", "lambda:InvokeFunction", "lambda:GetFunction", "lambda:ListAliases", "lambda:UpdateFunctionConfiguration", "lambda:UpdateFunctionCode", "lambda:AddPermission", "lambda:DeleteFunctionConcurrency", "lambda:RemovePermission", "lambda:GetPolicy" "lambda:GetAccountSettings", "lambda:ListFunctions", "lambda:ListEventSourceMappings",`	Autoriser la création et la gestion des fonctions Lambda répertoriées en tant que ressources. Dans l'exemple, un préfixe de nom est utilisé dans l'identifiant de ressource`arn:aws:lambda:*:` *MyAWSAcctId* `:function:` *MyAthenaLambdaFunctionsPrefix* `*`, tandis qu'un préfixe partagé *MyAthenaLambdaFunctionsPrefix* est utilisé dans le nom d'un groupe de fonctions Lambda afin qu'elles n'aient pas besoin d'être spécifiées individuellement en tant que ressources. Vous pouvez spécifier une ou plusieurs ressources de fonction Lambda.
`"s3:GetObject"`	Permet la lecture d'un bucket qui AWS Serverless Application Repository nécessite les informations spécifiées par l'identifiant de ressource`arn:aws:s3:::awsserverlessrepo-changesets-` *1iiv3xa62ln3m* `/*`. Ce compartiment peut être spécifique à votre compte.
`"cloudformation:*"`	Permet la création et la gestion des AWS CloudFormation piles spécifiées par la ressource *StackPrefixMyCF* . Ces piles et ensembles de piles permettent de AWS Serverless Application Repository déployer les connecteurs et les UDF.

Actions autorisées	Explication
`"serverlessrepo:*"`	Permet de rechercher, d'afficher, de publier et de mettre à jour des applications dans le AWS Serverless Application Repository, spécifié par l'identifiant de ressource`arn:aws:serverlessrepo:*:*:applications/*`.

Exemple de politiques d'autorisations IAM pour autoriser les fonctions définies par l'utilisateur (UDF) Amazon Athena

Les exemples de politique d'autorisation présentés dans cette rubrique illustrent les actions autorisées requises et les ressources pour lesquelles elles sont autorisées. Examinez attentivement ces politiques et modifiez-les en fonction de vos besoins avant d'attacher des politiques d'autorisation similaires aux identités IAM.

- Example Policy to Allow an IAM Principal to Run and Return Queries that Contain an Athena UDF Statement
- Example Policy to Allow an IAM Principal to Create an Athena UDF

Example – Autoriser un principal IAM à exécuter et renvoyer des requêtes contenant une instruction UDF Athena

La politique d'autorisations basée sur l'identité suivante autorise les actions dont un utilisateur ou un autre principal IAM a besoin pour exécuter des requêtes utilisant des instructions UDF Athena.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "athena:StartQueryExecution",
                "lambda:InvokeFunction",
                "athena:GetQueryResults",
                "s3:ListMultipartUploadParts",
```

```
                "athena:GetWorkGroup",
                "s3:PutObject",
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "athena:StopQueryExecution",
                "athena:GetQueryExecution",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:athena:*:MyAWSAcctId:workgroup/MyAthenaWorkGroup",
                "arn:aws:s3:::MyQueryResultsBucket/*",
                "arn:aws:lambda:*:MyAWSAcctId:function:OneAthenaLambdaFunction",
                "arn:aws:lambda:*:MyAWSAcctId:function:AnotherAthenaLambdaFunction"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "athena:ListWorkGroups",
            "Resource": "*"
        }
    ]
}
```

Explication des autorisations

Actions autorisées	Explication
`"athena:StartQueryExecution", "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StopQueryExecution", "athena:GetQueryExecution",`	Autorisations Athena nécessaires pour exécuter des requêtes dans le groupe de travail `MyAthenaWorkGroup` .
`"s3:PutObject", "s3:GetObject", "s3:AbortMultipartUpload"`	`s3:PutObject` et `s3:AbortMultipartUpload` autorisez l'écriture des résultats de la requête dans tous les sous-dossiers du bucket de résultats de requête tel que spécifié par l'identifiant de `arn:aws:s3::: MyQueryResultsBucket /*` ressource, où se *MyQueryResultsBuck*

Actions autorisées	Explication
	et trouve le bucket de résultats de requête Athena. Pour plus d'informations, consultez Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie. `s3:GetObject` permet de lire les résultats de la requête et l'historique des requêtes pour la ressource spécifiée sous la forme`arn:aws:s3::: MyQueryResultsBucket` , où se *MyQueryResultsBucket* trouve le bucket de résultats de requête Athena. Pour plus d'informations, consultez Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie. `s3:GetObject` permet également de lire à partir de la ressource spécifiée sous la forme`"arn:aws:s3::: MyLambdaSpillBucket /MyLambdaSpillPrefix *"`, où *MyLambdaSpillPrefix* est spécifiée dans la configuration de la ou des fonctions Lambda invoquées.
`"lambda:InvokeFunction"`	Permet aux requêtes d'invoquer les AWS Lambda fonctions spécifiées dans le `Resource` bloc. Par exemple`arn:aws:lambda:*: MyAWSAcctId :function : MyAthenaLambdaFunction` , où *MyAthenaLambdaFunction* indique le nom d'une fonction Lambda à invoquer. Plusieurs fonctions peuvent être spécifiées comme indiqué dans l'exemple.

Example – Autoriser un principal IAM à créer une UDF Athena

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "lambda:CreateFunction",
                "lambda:ListVersionsByFunction",
                "iam:CreateRole",
                "lambda:GetFunctionConfiguration",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "lambda:PutFunctionConcurrency",
                "iam:PassRole",
                "iam:DetachRolePolicy",
                "lambda:ListTags",
                "iam:ListAttachedRolePolicies",
                "iam:DeleteRolePolicy",
                "lambda:DeleteFunction",
                "lambda:GetAlias",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetPolicy",
                "lambda:InvokeFunction",
                "lambda:GetFunction",
                "lambda:ListAliases",
                "lambda:UpdateFunctionConfiguration",
                "iam:DeleteRole",
                "lambda:UpdateFunctionCode",
                "s3:GetObject",
                "lambda:AddPermission",
                "iam:UpdateRole",
                "lambda:DeleteFunctionConcurrency",
                "lambda:RemovePermission",
                "iam:GetRolePolicy",
                "lambda:GetPolicy"
            ],
            "Resource": [

 "arn:aws:lambda:*:111122223333:function:MyAthenaLambdaFunctionsPrefix*",
                "arn:aws:s3:::awsserverlessrepo-changesets-1iiv3xa62ln3m/*",
```

```
                "arn:aws:iam::*:role/RoleName",
                "arn:aws:iam::111122223333:policy/*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateUploadBucket",
                "cloudformation:DescribeStackDriftDetectionStatus",
                "cloudformation:ListExports",
                "cloudformation:ListStacks",
                "cloudformation:ListImports",
                "lambda:ListFunctions",
                "iam:ListRoles",
                "lambda:GetAccountSettings",
                "ec2:DescribeSecurityGroups",
                "cloudformation:EstimateTemplateCost",
                "ec2:DescribeVpcs",
                "lambda:ListEventSourceMappings",
                "cloudformation:DescribeAccountLimits",
                "ec2:DescribeSubnets",
                "cloudformation:CreateStackSet",
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "cloudformation:*",
            "Resource": [
                "arn:aws:cloudformation:*:111122223333:stack/aws-serverless-
repository-MyCFStackPrefix*/*",
                "arn:aws:cloudformation:*:111122223333:stack/
serverlessrepo-MyCFStackPrefix*/*",
                "arn:aws:cloudformation:*:*:transform/Serverless-*",
                "arn:aws:cloudformation:*:111122223333:stackset/aws-serverless-
repository-MyCFStackPrefix*:*",
                "arn:aws:cloudformation:*:111122223333:stackset/
serverlessrepo-MyCFStackPrefix*:*"
            ]
        },
        {
```

```
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "serverlessrepo:*",
            "Resource": "arn:aws:serverlessrepo:*:*:applications/*"
        }
    ]
}
```

Explication des autorisations

Actions autorisées	Explication
`"lambda:CreateFunction", "lambda:ListVersionsByFunction", "lambda:GetFunctionConfiguration", "lambda:PutFunctionConcurrency", "lambda:ListTags", "lambda:DeleteFunction", "lambda:GetAlias", "lambda:InvokeFunction", "lambda:GetFunction", "lambda:ListAliases", "lambda:UpdateFunctionConfiguration", "lambda:UpdateFunctionCode", "lambda:AddPermission", "lambda:DeleteFunctionConcurrency", "lambda:RemovePermission", "lambda:GetPolicy" "lambda:GetAccountSettings", "lambda:ListFunctions", "lambda:ListEventSourceMappings",`	Autoriser la création et la gestion des fonctions Lambda répertoriées en tant que ressources. Dans l'exemple, un préfixe de nom est utilisé dans l'identifiant de ressource`arn:aws:lambda:*:` *MyAWSAcctId* `:function:` *MyAthenaLambdaFunctionsPrefix* `*`, tandis qu'un préfixe partagé *MyAthenaLambdaFunctionsPrefix* est utilisé dans le nom d'un groupe de fonctions Lambda afin qu'elles n'aient pas besoin d'être spécifiées individuellement en tant que ressources. Vous pouvez spécifier une ou plusieurs ressources de fonction Lambda.
`"s3:GetObject"`	Permet la lecture d'un bucket qui AWS Serverless Application Repository nécessite les informations spécifiées par l'identifiant de ressource`arn:aws:s3:::awsserverlessrepo-changesets-` *1iiv3xa62ln3m* `/*`.

Actions autorisées	Explication
`"cloudformation:*"`	Permet la création et la gestion des AWS CloudFormation piles spécifiées par la ressource *`StackPrefixMyCF`* . Ces piles et ensembles de piles permettent de AWS Serverless Application Repository déployer les connecteurs et les UDF.
`"serverlessrepo:*"`	Permet de rechercher, d'afficher, de publier et de mettre à jour des applications dans le AWS Serverless Application Repository, spécifié par l'identifiant de ressource`arn:aws:serverlessrepo:*:*:applications/*`.

Autorisation d'accès pour ML avec Athena

Les principals IAM qui exécutent des requêtes Athena ML doivent être autorisés à effectuer l'action `sagemaker:invokeEndpoint` pour les points de terminaison Sagemaker qu'ils utilisent. Incluez une déclaration de politique similaire à la suivante dans les politiques d'autorisations basées sur l'identité attachées aux identités utilisateur. En outre, attachez la politique AWS politique gérée : AmazonAthenaFullAccess, qui accorde un accès complet aux actions Athena, ou une politique en ligne modifiée qui autorise un sous-ensemble d'actions.

Remplacez `arn:aws:sagemaker:`*`region`*`:`*`AWSAcctID`*`:`*`ModelEndpoint`* dans l'exemple par l'ARN ou les ARN des points de terminaison du modèle à utiliser dans les requêtes. Pour plus d'informations, consultez la section Actions, ressources et clés de condition SageMaker dans la référence d'autorisation de service.

```
{
            "Effect": "Allow",
            "Action": [
                "sagemaker:invokeEndpoint"
            ],
            "Resource": "arn:aws:sagemaker:us-west-2:123456789012:workteam/public-
crowd/default"
}
```

Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Activation de l'accès fédéré à l'API Athena

Cette section décrit l'accès fédéré qui autorise un utilisateur ou une application cliente de votre organisation à appeler des opérations d'API Amazon Athena. Dans ce cas, les utilisateurs de l'organisation ne disposent pas d'un accès direct à Athena. Au lieu de cela, vous gérez les informations d'identification des utilisateurs AWS en dehors de Microsoft Active Directory. Active Directory prend en charge SAML 2.0 (Security Assertion Markup Language 2.0).

Pour authentifier les utilisateurs, utilisez le pilote JDBC ou ODBC avec la prise en charge de SAML 2.0 pour accéder à Active Directory Federation Services (AD FS) 3.0 et permettre à une application cliente d'appeler les opérations d'API Athena.

Pour plus d'informations sur la prise en charge de SAML 2.0 AWS, voir À propos de la fédération SAML 2.0 dans le guide de l'utilisateur IAM.

> **Note**
>
> L'accès fédéré à l'API Athena est pris en charge pour un type de fournisseur d'identité (IdP) spécifique, l'Active Directory Federation Service (ADFS 3.0), qui fait partie de Windows Server. L'accès fédéré n'est pas compatible avec la fonctionnalité de propagation d'identité approuvée d'IAM Identity Center. L'accès est établi via les versions des pilotes JDBC ou ODBC prenant en charge SAML 2.0. Pour plus d'informations, consultez Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

Rubriques

- Avant de commencer
- Diagramme d'architecture
- Procédure : accès fédéré SAML à l'API Athena

Avant de commencer

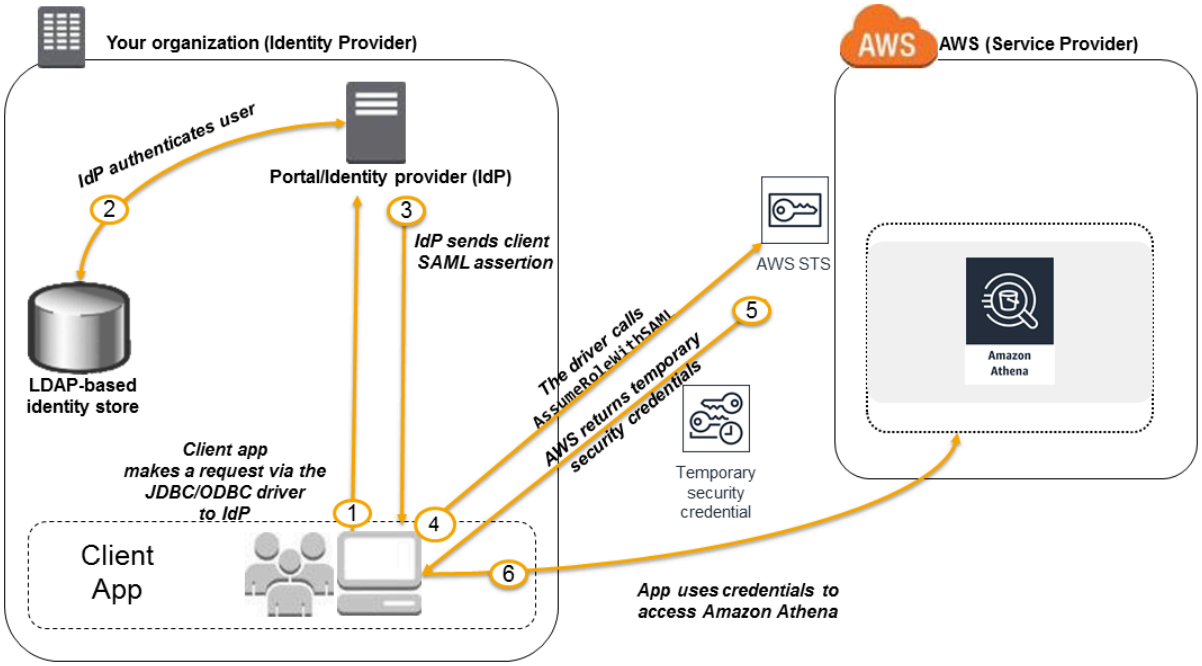
Avant de commencer, effectuez les opérations obligatoires suivantes :

- Dans votre organisation, installez et configurez AD FS 3.0 comme votre fournisseur d'identité.

- Installez et configurez les dernières versions disponibles des pilotes JDBC ou ODBC sur les clients qui sont utilisés pour l'accès à Athena. Le pilote doit inclure la prise en charge de l'accès fédéré compatible avec SAML 2.0. Pour plus d'informations, consultez Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

Diagramme d'architecture

Le schéma suivant illustre ce processus.



1. A l'aide d'une application cliente dotée du pilote JDBC ou ODBC, un utilisateur de l'organisation demande son authentification par l'IdP de l'organisation. L'IdP est AD FS 3.0.
2. L'IdP authentifie l'utilisateur par rapport à Active Directory, qui est la base d'identités de l'organisation.
3. L'IdP crée une assertion SAML à l'aide des informations concernant l'utilisateur et l'envoie à l'application cliente via le pilote JDBC ou ODBC.
4. Le pilote JDBC ou ODBC appelle l'opération d'API AWS Security Token Service AssumeRoleWithSAML en lui transmettant les paramètres suivants :
   - L'ARN du fournisseur SAML
   - ARN du rôle à assumer.

- Assertion SAML de l'IdP

  Pour plus d'informations, consultez AssumeRoleWithSAML dans la référence de l'AWS Security Token Service API.

5. La réponse de l'API à l'application cliente via le pilote JDBC ou ODBC inclut les informations d'identification de sécurité temporaires.
6. L'application cliente utilise ces informations d'identification de sécurité temporaires pour appeler les opérations d'API Athena, ce qui permet aux utilisateurs d'accéder aux opérations d'API Athena.

Procédure : accès fédéré SAML à l'API Athena

Cette procédure établit un lien de confiance entre l'IdP de votre organisation et votre AWS compte afin de permettre un accès fédéré basé sur SAML au fonctionnement de l'API Amazon Athena.

Pour activer l'accès fédéré à l'API Athena :

1. Dans votre organisation, enregistrez-vous AWS en tant que fournisseur de services (SP) auprès de votre IdP. Ce processus est connu sous le nom d'approbation des parties utilisatrices. Pour plus d'informations, consultez la rubrique Configuration de votre IdP SAML 2.0 à l'aide d'une relation d'approbation des parties utilisatrices du Guide de l'utilisateur IAM. Dans le cadre de cette tâche, effectuez les étapes suivantes :

   a. Obtenez l'exemple de document de métadonnées SAML à partir de cette URL : https://signin.aws.amazon.com/static/saml-metadata.xml.

   b. Dans l'IdP (ADFS) de votre organisation, générez un fichier XML de métadonnées équivalent qui décrit votre IdP en tant que fournisseur d'identité pour. AWS Votre fichier de métadonnées doit inclure le nom de l'émetteur, la date de création, la date d'expiration et les clés AWS utilisées pour valider les réponses d'authentification (assertions) de votre organisation.
2. Dans la console IAM, créez une entité de fournisseur d'identité SAML. Pour plus d'informations, consultez la rubrique Création de fournisseurs d'identité SAML du Guide de l'utilisateur IAM. Dans le cadre de cette étape, effectuez ce qui suit :

   a. Ouvrez la console IAM à l’adresse https://console.aws.amazon.com/iam/.

   b. Chargez le document de métadonnées SAML généré par l'IdP (AD FS) à l'étape 1 de cette procédure.

3. Dans la console IAM, créez un ou plusieurs rôles IAM pour votre IdP. Pour plus d'informations, consultez la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) du Guide de l'utilisateur IAM. Dans le cadre de cette étape, effectuez ce qui suit :

   - Dans la politique d'autorisation du rôle, dressez une liste des actions que les utilisateurs de l'organisation sont autorisés à effectuer dans AWS.
   - Dans la politique d'approbation du rôle, définissez le fournisseur d'identité que vous avez créée à l'étape 2 de cette procédure en tant que principal.

   Cela établit une relation de confiance entre votre organisation et AWS.

4. Dans l'IdP de votre organisation (ADFS), vous définissez les assertions qui associent les utilisateurs et les groupes de l'organisation aux rôles IAM. Le mappage des utilisateurs et des groupes aux rôles IAM est également connu sous le nom de règle de demande. Notez que différents utilisateurs et groupes de l'organisation peuvent être associés à différents rôles IAM.

   Pour plus d'informations sur la configuration du mappage dans ADFS, consultez le billet de blog : Activation de la fédération à AWS l'aide de Windows Active Directory, ADFS et SAML 2.0.

5. Installez et configurez le pilote JDBC ou ODBC avec la prise en charge de SAML 2.0. Pour plus d'informations, consultez Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

6. Spécifiez la chaîne de connexion de votre application vers le pilote JDBC ou ODBC. Pour plus d'informations sur la chaîne de connexion que votre application doit utiliser, consultez la section « Utilisation du fournisseur d'informations d'identification d'Active Directory Federation Services (ADFS) » du Guide de configuration et d'installation du pilote JDBC, ou une rubrique similaire du Guide de configuration et d'installation du pilote ODBC disponible au téléchargement au format PDF à partir des rubriques Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

   Le résumé global de la configuration de la chaîne de connexion pour les pilotes est le suivant :

   1. Dans `AwsCredentialsProviderClass configuration`, définissez le `com.simba.athena.iamsupport.plugin.AdfsCredentialsProvider` pour indiquer que vous souhaitez utiliser l'authentification basée sur SAML 2.0 via l'IdP AD FS.
   2. Pour `idp_host`, fournissez le nom d'hôte du serveur d'IdP AD FS.
   3. Pour `idp_port`, fournissez le numéro de port que l'IdP AD FS écoute pour la demande d'assertion SAML.

4. Pour `UID` et `PWD`, fournissez les informations d'identification de l'utilisateur de domaine AD. Lorsque vous utilisez le pilote sur Windows, si `UID` et `PWD` ne sont pas fournis, le pilote tente d'obtenir les informations d'identification de l'utilisateur connecté à la machine Windows.
5. Vous pouvez également définir `ssl_insecure` sur `true`. Dans ce cas, le pilote ne vérifie pas l'authenticité du certificat SSL du serveur de l'IdP AD FS. Le fait de définir sur `true` est nécessaire si le certificat SSL de l'IdP AD FS n'a pas été configuré pour être approuvé par le pilote.
6. Pour activer le mappage d'un utilisateur ou d'un groupe du domaine Active Directory utilisateur à un ou plusieurs rôles IAM (comme indiqué à l'étape 4 de cette procédure), dans le `preferred_role` pour la connexion JDBC ou ODBC, spécifiez le rôle IAM (ARN) à endosser pour le pilote de connexion. La spécification du `preferred_role` est facultative, et elle est utile si le rôle n'est pas le premier répertorié dans la règle de demande.

En conséquence de cette procédure, les actions suivantes se produisent :

1. Le pilote JDBC ou ODBC appelle l'API AWS STSAssumeRoleWithSAML et lui transmet les assertions, comme indiqué à l'étape 4 du schéma d'architecture.
2. AWS s'assure que la demande pour assumer le rôle provient de l'IdP référencé dans l'entité fournisseur SAML.
3. Si la demande aboutit, l'opération d'API AWS STS AssumeRoleWithSAML renvoie un ensemble d'informations d'identification de sécurité temporaires, que votre application cliente utilise pour envoyer des demandes signées à Athena.

   Votre application dispose maintenant d'informations sur l'utilisateur actuel et peut accéder à Athena par programmation.

Journalisation et surveillance dans Athena

Pour détecter les incidents, recevoir des alertes en cas d'incident et y répondre, utilisez ces options avec Amazon Athena :

- Surveiller Athéna avec AWS CloudTrail : AWS CloudTrailfournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un utilisateur dans Service AWS Athéna. Cette fonction capture les appels à partir de la console Athena et les appels de code aux opérations d'API Athena en tant qu'événements. Cela vous permet de déterminer la demande qui a été faite à Athena, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été

faite, ainsi que d'autres détails. Pour plus d'informations, consultez Journalisation des appels d'API Amazon Athena avec AWS CloudTrail.

Vous pouvez également utiliser Athena pour interroger les fichiers CloudTrail journaux non seulement pour Athéna, mais aussi pour d'autres. Services AWS Pour plus d'informations, consultez Journaux d'interrogation AWS CloudTrail.

- Surveillez l'utilisation d'Athena avec et CloudTrail Amazon — QuickSight Amazon QuickSight est un service de business intelligence entièrement géré et basé sur le cloud qui vous permet de créer des tableaux de bord interactifs auxquels votre organisation peut accéder depuis n'importe quel appareil. Pour un exemple de solution qui utilise CloudTrail Amazon pour surveiller l'utilisation QuickSight d'Athena, consultez le billet de blog AWS Big Data How Realtor.com monitors Amazon Athena utilisation with et Amazon. AWS CloudTrail QuickSight
- Utilisation EventBridge avec Athena : Amazon EventBridge fournit un flux d'événements système en temps quasi réel qui décrivent les modifications apportées aux AWS ressources. EventBridge prend connaissance des changements opérationnels au fur et à mesure qu'ils se produisent, y répond et prend les mesures correctives nécessaires, en envoyant des messages pour répondre à l'environnement, en activant des fonctions, en apportant des modifications et en capturant des informations d'état. Les événements sont générés dans la mesure du possible. Pour plus d'informations, consultez Getting started with Amazon EventBridge dans le guide de EventBridge l'utilisateur Amazon.
- Utilisez des groupes de travail pour séparer les utilisateurs, les équipes, les applications ou les charges de travail, ainsi que pour définir des limites de requêtes et contrôler les coûts des requêtes. Vous pouvez consulter les métriques liées aux requêtes dans Amazon CloudWatch, contrôler les coûts des requêtes en configurant des limites à la quantité de données numérisées, créer des seuils et déclencher des actions, telles que des alarmes Amazon SNS, lorsque ces seuils sont dépassés. Pour une procédure de haut niveau, consultez Configuration de groupes de travail. Utilisation des autorisations IAM au niveau des ressources pour contrôler l'accès à un groupe de travail spécifique. Pour plus d'informations, consultez Utilisation de groupes de travail pour exécuter des requêtes et Contrôle des coûts et surveillance des requêtes à l'aide de CloudWatch métriques et d'événements.

Rubriques

- Journalisation des appels d'API Amazon Athena avec AWS CloudTrail

Journalisation des appels d'API Amazon Athena avec AWS CloudTrail

Athena est intégrée à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un dans Service AWS Athena.

CloudTrail capture tous les appels d'API pour Athena sous forme d'événements. Les appels capturés incluent les appels de la console Athena et les appels de code adressés aux opérations d'API Athena. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris des événements pour Athena. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements.

À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à Athena, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des informations supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Vous pouvez utiliser Athena pour interroger les fichiers CloudTrail journaux d'Athéna elle-même et d'autres. Services AWS Pour plus d'informations, consultez Journaux d'interrogation AWS CloudTrailHive JSON SerDe, et le billet de blog sur le AWS Big Data Utiliser les déclarations CTAS avec Amazon Athena pour réduire les coûts et améliorer les performances, qui fournit des informations sur l' CloudTrail utilisation d'Athena.

Informations sur Athéna dans CloudTrail

CloudTrail est activé sur votre compte Amazon Web Services lorsque vous créez le compte. Lorsqu'une activité se produit dans Athena, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez afficher, rechercher et télécharger les événements récents dans votre compte Amazon Web Services. Pour plus d'informations, consultez la section Affichage des événements avec l'historique des CloudTrail événements.

Pour un registre continu des événements dans votre compte Amazon Web Services, y compris les événements pour Athena, créez un journal d'activité. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d’activité dans la console, il s’applique à toutes les régions Régions AWS. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez en configurer d'autres Services

AWS pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d'informations, consultez les ressources suivantes :

- Présentation de la création d'un journal de suivi
- CloudTrail services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Toutes les actions d'Athena sont enregistrées CloudTrail et documentées dans le manuel Amazon Athena API Reference. Par exemple, les appels aux GetQueryResultsactions StartQueryExecutionet génèrent des entrées dans les fichiers CloudTrail journaux.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec les informations d'identification de l'utilisateur root ou AWS Identity and Access Management (IAM).
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la requête a été effectuée par un autre Service AWS.

Pour plus d'informations, consultez l'élément CloudTrail UserIdentity.

Présentation des entrées du fichier journal Athena

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

> **Note**
>
> Pour empêcher la divulgation involontaire d'informations sensibles, l'entrée `queryString` dans les journaux `StartQueryExecution` et `CreateNamedQuery` a une valeur de `***OMITTED***`. Ce comportement est intégré à la conception. Pour accéder à la chaîne de requête réelle, vous pouvez utiliser l'GetQueryExecutionAPI Athena et transmettre la valeur de `responseElements.queryExecutionId` depuis le CloudTrail journal.

Les exemples suivants illustrent les entrées de CloudTrail journal pour :

- StartQueryExecution(Succès)
- StartQueryExecution (Échec)
- CreateNamedQuery

StartQueryExecution (réussite)


```
{
 "eventVersion":"1.05",
 "userIdentity":{
    "type":"IAMUser",
    "principalId":"EXAMPLE_PRINCIPAL_ID",
    "arn":"arn:aws:iam::123456789012:user/johndoe",
    "accountId":"123456789012",
    "accessKeyId":"EXAMPLE_KEY_ID",
    "userName":"johndoe"
 },
 "eventTime":"2017-05-04T00:23:55Z",
 "eventSource":"athena.amazonaws.com",
 "eventName":"StartQueryExecution",
 "awsRegion":"us-east-1",
 "sourceIPAddress":"77.88.999.69",
 "userAgent":"aws-internal/3",
 "requestParameters":{
    "clientRequestToken":"16bc6e70-f972-4260-b18a-db1b623cb35c",
    "resultConfiguration":{
       "outputLocation":"s3://DOC-EXAMPLE-BUCKET/test/"
    },
    "queryString":"***OMITTED***"
 },
```

```
 "responseElements":{
    "queryExecutionId":"b621c254-74e0-48e3-9630-78ed857782f9"
 },
 "requestID":"f5039b01-305f-11e7-b146-c3fc56a7dc7a",
 "eventID":"c97cf8c8-6112-467a-8777-53bb38f83fd5",
 "eventType":"AwsApiCall",
 "recipientAccountId":"123456789012"
}
```

StartQueryExecution (échec)

```
{
 "eventVersion":"1.05",
 "userIdentity":{
  "type":"IAMUser",
  "principalId":"EXAMPLE_PRINCIPAL_ID",
  "arn":"arn:aws:iam::123456789012:user/johndoe",
  "accountId":"123456789012",
  "accessKeyId":"EXAMPLE_KEY_ID",
  "userName":"johndoe"
  },
 "eventTime":"2017-05-04T00:21:57Z",
 "eventSource":"athena.amazonaws.com",
 "eventName":"StartQueryExecution",
 "awsRegion":"us-east-1",
 "sourceIPAddress":"77.88.999.69",
 "userAgent":"aws-internal/3",
 "errorCode":"InvalidRequestException",
 "errorMessage":"Invalid result configuration. Should specify either output location or
 result configuration",
 "requestParameters":{
  "clientRequestToken":"ca0e965f-d6d8-4277-8257-814a57f57446",
  "queryString":"***OMITTED***"
  },
 "responseElements":null,
 "requestID":"aefbc057-305f-11e7-9f39-bbc56d5d161e",
 "eventID":"6e1fc69b-d076-477e-8dec-024ee51488c4",
 "eventType":"AwsApiCall",
 "recipientAccountId":"123456789012"
}
```

CreateNamedQuery


```
{
  "eventVersion":"1.05",
  "userIdentity":{
     "type":"IAMUser",
     "principalId":"EXAMPLE_PRINCIPAL_ID",
     "arn":"arn:aws:iam::123456789012:user/johndoe",
     "accountId":"123456789012",
     "accessKeyId":"EXAMPLE_KEY_ID",
     "userName":"johndoe"
  },
  "eventTime":"2017-05-16T22:00:58Z",
  "eventSource":"athena.amazonaws.com",
  "eventName":"CreateNamedQuery",
  "awsRegion":"us-west-2",
  "sourceIPAddress":"77.88.999.69",
  "userAgent":"aws-cli/1.11.85 Python/2.7.10 Darwin/16.6.0 botocore/1.5.48",
  "requestParameters":{
     "name":"johndoetest",
     "queryString":"***OMITTED***",
     "database":"default",
     "clientRequestToken":"fc1ad880-69ee-4df0-bb0f-1770d9a539b1"
     },
  "responseElements":{
     "namedQueryId":"cdd0fe29-4787-4263-9188-a9c8db29f2d6"
     },
  "requestID":"2487dd96-3a83-11e7-8f67-c9de5ac76512",
  "eventID":"15e3d3b5-6c3b-4c7c-bc0b-36a8dd95227b",
  "eventType":"AwsApiCall",
  "recipientAccountId":"123456789012"
},
```


Validation de la conformité pour Amazon Athena

Des auditeurs tiers évaluent la sécurité et la conformité d'Amazon Athena Gateway dans le cadre de plusieurs programmes de conformité AWS. Il s'agit notamment des certifications SOC, PCI, FedRAMP, et autres.

Pour obtenir la liste des Services AWS concernés par des programmes de conformité spécifiques, consultez Services AWS concernés par les programmes de conformité. Pour obtenir des informations générales, veuillez consulter Programmes de conformité d'AWS.

Vous pouvez télécharger les rapports de l'audit externe avec AWS Artifact. Pour plus d'informations, veuillez consulter Téléchargement des rapports dans AWS Artifact.

Votre responsabilité en matière de conformité lorsque vous utilisez Athena est déterminée par la sensibilité de vos données, des objectifs de conformité de votre entreprise, ainsi que de la législation et de la réglementation en vigueur. AWS fournit les ressources suivantes pour faciliter le respect de la conformité :

- Guides de démarrage rapide de la sécurité et de la conformité – Ces guides de déploiement traitent de considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de référence centrés sur la sécurité et la conformité dans AWS.
- Architecture pour la sécurité et la conformité HIPAA sur Amazon Web Services : ce livre blanc décrit comment les entreprises peuvent utiliser AWS pour créer des applications conformes à la loi HIPAA.
- Ressources de conformité d'AWS – Cet ensemble de manuels et de guides peut s'appliquer à votre secteur et à votre emplacement.
- AWS Config – Ce Service AWS permet d'évaluer la conformité des configurations de vos ressources par rapport à des pratiques internes, règlementations et autres directives sectorielles.
- AWS Security Hub – Ce Service AWS fournit une vue complète de votre état de sécurité au sein d'AWS, ce qui vous permet de vérifier votre conformité aux normes du secteur et aux bonnes pratiques de sécurité.

Résilience dans Athena

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité. Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur les zones de disponibilité Régions AWS et les zones de disponibilité, consultez la section Infrastructure AWS globale.

Outre l'infrastructure AWS mondiale, Athena propose plusieurs fonctionnalités pour répondre à vos besoins en matière de résilience et de sauvegarde des données.

Comme Athena est sans serveur, il n'y a aucune infrastructure à configurer ou à gérer. Athena est hautement disponible et exécute les requêtes à l'aide de ressources de calcul sur plusieurs zones de disponibilité, acheminant automatiquement les requêtes de manière appropriée si une zone de disponibilité particulière est inaccessible. Athena utilise Simple Storage Service (Amazon S3) pour le stockage sous-jacent des données, ce qui rend vos données hautement disponibles et durables. Simple Storage Service (Amazon S3) fournit une infrastructure durable pour stocker des données importantes et est conçu pour offrir une durabilité de 99,999999999 % des objets. Vos données sont stockées de manière redondante sur plusieurs installations et sur plusieurs appareils au sein de chaque installation.

Sécurité de l'infrastructure dans Athena

En tant que service géré, Amazon Athena est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Athena via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Utilisation des politiques IAM pour restreindre l'accès aux opérations Athena. Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Les politiques gérées par Athena sont faciles à utiliser et sont automatiquement mises à jour avec les actions requises à mesure que le service évolue. Les politiques gérées par le client et les politiques en ligne vous permettent d'affiner les politiques en spécifiant des actions Athena plus détaillées au

sein de la politique. Octroi d'un accès approprié à l'emplacement Simple Storage Service (Amazon S3) des données. Pour obtenir des informations détaillées et des scénarios sur la manière d'accorder l'accès à Simple Storage Service (Amazon S3), consultez la rubrique Exemples : gestion de l'accès du Guide du développeur Amazon Simple Storage Service. Pour plus d'informations et un exemple des actions Simple Storage Service (Amazon S3) à autoriser, consultez l'exemple de politique de compartiment dans la rubrique Accès intercompte.

Rubriques

- Connexion à Amazon Athena à l'aide d'un point de terminaison de VPC d'interface

Connexion à Amazon Athena à l'aide d'un point de terminaison de VPC d'interface

Vous pouvez renforcer la sécurité de votre VPC en utilisant un point de terminaison d'un VPC d'interface (AWS PrivateLink) et un point de terminaison d'un VPC AWS Glue dans votre cloud privé virtuel (VPC). Un point de terminaison d'un VPC d'interface améliore la sécurité en vous permettant de contrôler les destinations accessibles depuis l'intérieur de votre VPC. Chaque point de terminaison d'un VPC est représenté par une ou plusieurs interfaces réseau Elastic (ENI) avec des adresses IP privées dans vos sous-réseaux VPC.

Le point de terminaison VPC de l'interface connecte votre VPC directement à Athena sans passerelle Internet, périphérique NAT, connexion VPN ou connexion. AWS Direct Connect Les instances de votre VPC ne nécessitent pas d'adresses IP publiques pour communiquer avec l'API Athena.

Pour utiliser Athena via votre VPC, vous devez vous connecter à partir d'une instance située dans le VPC ou connecter votre réseau privé à votre VPC à l'aide d'un réseau privé virtuel (VPN) Amazon ou du AWS Direct Connect. Pour obtenir des informations sur Amazon VPN, consultez la rubrique Connexions VPN du Guide de l'utilisateur Amazon Virtual Private Cloud. Pour plus d'informations AWS Direct Connect, voir Création d'une connexion dans le Guide de AWS Direct Connect l'utilisateur.

Athena prend en charge les points de terminaison VPC partout où Régions AWS Amazon VPC et Athena sont disponibles.

Vous pouvez créer un point de terminaison VPC d'interface pour vous connecter à Athena à l'aide des commandes AWS Management Console or AWS Command Line Interface ().AWS CLI Pour plus d'informations, consultez Création d'un point de terminaison d'interface.

Une fois que vous avez créé un point de terminaison de VPC d'interface, si vous activez les noms d'hôte DNS privés pour le point de terminaison, le point de terminaison Athena par défaut (https://athena.*Region*.amazonaws.com) est résolu par votre point de terminaison de VPC.

Si vous n'activez pas les noms d'hôte DNS privés, Amazon VPC fournit un nom de point de terminaison DNS que vous pouvez utiliser au format suivant :

```
VPC_Endpoint_ID.athena.Region.vpce.amazonaws.com
```

Pour plus d'informations, consultez la section Interface VPC endpoints (AWS PrivateLink) dans le guide de l'utilisateur Amazon VPC.

Athena prend en charge l'exécution d'appels en direction de toutes ses actions d'API à l'intérieur de votre VPC.

Création d'une politique de point de terminaison de VPC pour Athena

Vous pouvez créer une politique pour les points de terminaison de VPC Amazon pour Athena dans laquelle vous pouvez spécifier des restrictions telles que :

- Principal : le principal qui peut exécuter des actions.
- Actions : les actions qui peuvent être effectuées.
- Ressources : les ressources sur lesquelles les actions peuvent être exécutées.
- Identités fiables uniquement : utilisez `aws:PrincipalOrgId` cette condition pour restreindre l'accès aux seules informations d'identification appartenant à votre AWS organisation. Cela peut permettre d'empêcher l'accès par des principaux involontaires.
- Ressources approuvées uniquement : utilisez la condition `aws:ResourceOrgId` pour empêcher l'accès à des ressources involontaires.
- Identités et ressources approuvées uniquement : créez une politique combinée pour un point de terminaison de VPC afin d'empêcher l'accès à des principaux et à des ressources involontaires.

Pour plus d'informations, consultez la section Contrôle de l'accès aux services avec des points de terminaison VPC dans le guide de l'utilisateur Amazon VPC et l'annexe 2 — Exemples de politiques relatives aux points de terminaison VPC dans le livre blanc sur la AWS création d'un périmètre de données sur. AWS

Example – politique de point de terminaison de VPC

L'exemple suivant autorise les demandes par identité d'organisation aux ressources de l'organisation et autorise les demandes par les responsables AWS de service.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRequestsByOrgsIdentitiesToOrgsResources",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "my-org-id",
                    "aws:ResourceOrgID": "my-org-id"
                }
            }
        },
        {
            "Sid": "AllowRequestsByAWSServicePrincipals",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "aws:PrincipalIsAWSService": "true"
                }
            }
        }
    ]
}
```

Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Sous-réseaux partagés

Vous ne pouvez pas créer, décrire, modifier ou supprimer des points de terminaison d'un VPC dans des sous-réseaux qui sont partagés avec vous. Toutefois, vous pouvez utiliser les points de terminaison d'un VPC dans les sous-réseaux qui sont partagés avec vous. Pour plus d'informations sur le partage de VPC, consultez Partager votre VPC avec d'autres comptes dans le Guide de l'utilisateur Amazon VPC.

Configuration et analyse des vulnérabilités dans Athena

Athena fonctionne sans serveur, il n'y a donc aucune infrastructure à configurer ou à gérer. AWS gère les tâches de sécurité de base, telles que l'application de correctifs au système d'exploitation client (OS) et aux bases de données, la configuration du pare-feu et la reprise après sinistre. Ces procédures ont été vérifiées et certifiées par les tiers appropriés. Pour plus de détails, consultez les AWS ressources suivantes :

- Modèle de responsabilité partagée
- Bonnes pratiques en matière de sécurité, d'identité et de conformité

Utilisation d'Athena pour interroger des données enregistrées dans AWS Lake Formation

AWS Lake Formation vous permet de définir et d'appliquer des politiques d'accès au niveau des bases de données, des tables et des colonnes lorsque vous utilisez des requêtes Athena pour lire des données stockées dans Simple Storage Service (Amazon S3). Lake Formation fournit une couche d'autorisation et de gouvernance des données stockées dans Simple Storage Service (Amazon S3). Vous pouvez utiliser une hiérarchie d'autorisations dans Lake Formation pour accorder ou révoquer les autorisations de lecture des objets du catalogue de données tels que les bases de données, les tables et les colonnes. Lake Formation simplifie la gestion des autorisations et vous permet d'implémenter un contrôle d'accès précis (FGAC) pour vos données.

Vous pouvez utiliser Athena pour interroger à la fois les données qui sont enregistrées dans Lake Formation et celles qui ne le sont pas.

Les autorisations de Lake Formation s'appliquent lorsqu'on utilise Athena pour interroger des données sources à partir d'emplacements Simple Storage Service (Amazon S3) enregistrés dans Lake Formation. Les autorisations de Lake Formation s'appliquent également lorsque vous créez des

bases de données et des tables qui pointent vers des emplacements de données Simple Storage Service (Amazon S3) enregistrés. Pour utiliser Athena avec des données enregistrées à l'aide de Lake Formation, Athena doit être configuré pour utiliser le AWS Glue Data Catalog.

Les autorisations de Lake Formation ne s'appliquent pas lors de l'écriture d'objets dans Simple Storage Service (Amazon S3), ni lors de l'interrogation de données stockées dans Simple Storage Service (Amazon S3) ou de métadonnées qui ne sont pas enregistrées dans Lake Formation. Pour les données source dans Amazon S3 et les métadonnées qui ne sont pas enregistrées auprès de Lake Formation, l'accès est déterminé par les politiques d'autorisation IAM pour Amazon S3 et AWS Glue les actions. Les emplacements des résultats des requêtes Athena dans Simple Storage Service (Amazon S3) ne peuvent pas être enregistrés dans Lake Formation, et les politiques d'autorisation IAM pour Simple Storage Service (Amazon S3) contrôlent l'accès. En outre, les autorisations de Lake Formation ne s'appliquent pas à l'historique des requêtes Athena. Vous pouvez utiliser des groupes de travail Athena pour contrôler l'accès à l'historique des requêtes.

Pour plus d'informations sur Lake Formation, consultez la FAQ sur Lake Formation et le Guide du développeur AWS Lake Formation.

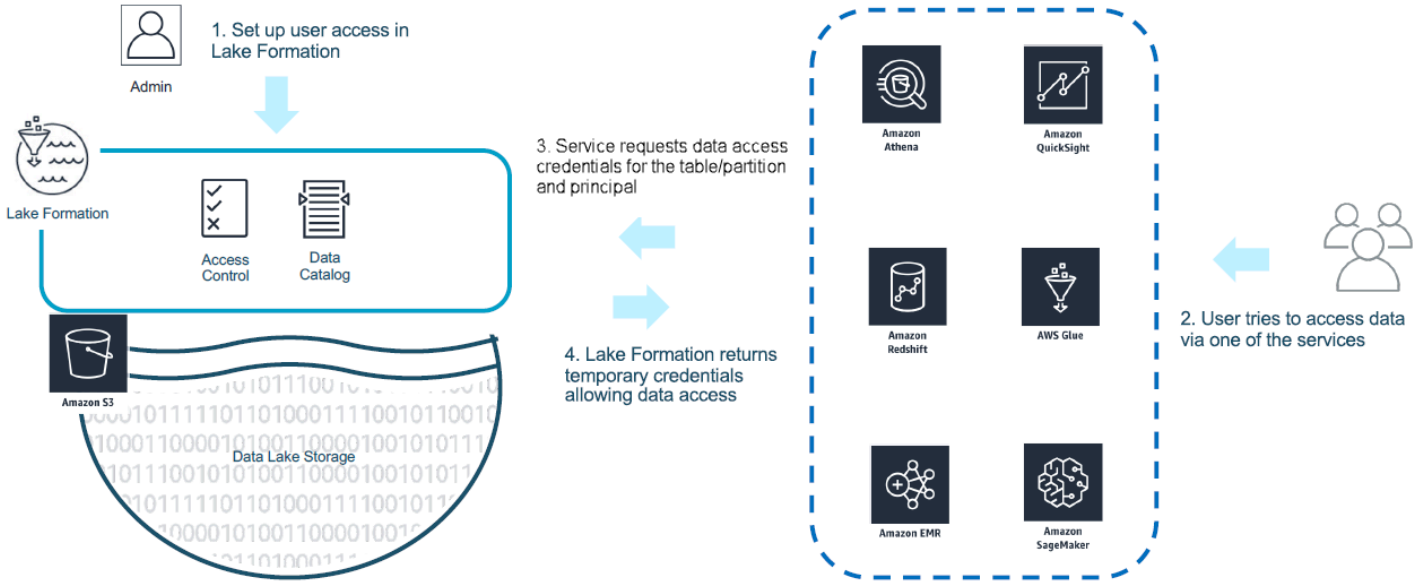
Rubriques

Mode d'accès d'Athena aux données enregistrées dans Lake Formation

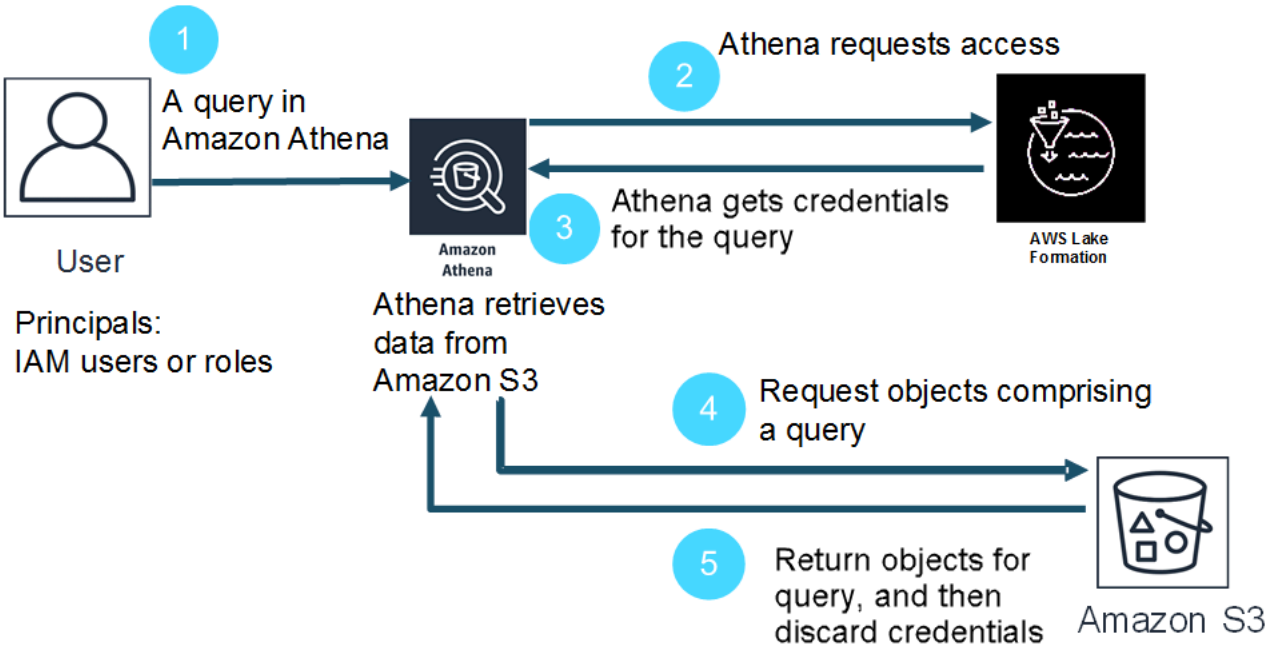
Le flux de travail d'accès décrit dans cette section s'applique uniquement à l'exécution de requêtes Athena sur des emplacements Simple Storage Service (Amazon S3) et des objets de métadonnées enregistrés dans Lake Formation. Pour plus d'informations, consultez la rubrique Enregistrement d'un lac de données du Guide du développeur AWS Lake Formation . Outre l'enregistrement des données, l'administrateur de Lake Formation applique les autorisations Lake Formation qui accordent ou révoquent l'accès aux métadonnées du catalogue de données et à l'emplacement des données dans Simple Storage Service (Amazon S3). Pour de plus amples informations, consultez Sécurité et contrôle d'accès aux métadonnées et données dans le Guide du développeur AWS Lake Formation .

Chaque fois qu'un principal Athena (utilisateur, groupe ou rôle) exécute une requête sur des données enregistrées à l'aide de Lake Formation, Lake Formation vérifie que le principal dispose des autorisations Lake Formation appropriées pour la base de données, la table et l'emplacement Simple Storage Service (Amazon S3), en fonction de la requête. Si le principal a accès, Lake Formation apporte des informations d'identification temporaires à Athena et la requête s'exécute.

Le schéma suivant illustre le flux décrit ci-dessus.



Le schéma suivant montre le fonctionnement de la vente d'identifiants à Athena sur la query-by-query base d'une `SELECT` requête hypothétique sur une table avec un site Amazon S3 enregistré dans Lake Formation :



1. Un principal exécute une requête SELECT dans Athena.
2. Athena analyse la requête et vérifie les autorisations de Lake Formation pour voir si le principal a été autorisé à accéder à la table et à ses colonnes.
3. Si le principal a accès, Athena demande des informations d'identification à Lake Formation. Si le principal n'a pas accès, Athena génère une erreur d'accès refusé.
4. Lake Formation fournit à Athena les informations d'identification à utiliser lors de la lecture des données à partir de Simple Storage Service (Amazon S3), ainsi que la liste des colonnes autorisées.
5. Athena utilise les informations d'identification temporaires de Lake Formation pour interroger les données de Simple Storage Service (Amazon S3). Une fois la requête terminée, Athena supprime les informations d'identification.

Considérations et limites de l'utilisation d'Athena pour l'interrogation de données enregistrées dans Lake Formation

Tenez compte des points suivants lorsque vous utilisez Athena pour interroger des données enregistrées dans Lake Formation. Pour plus d'informations, consultez la rubrique Problèmes connus liés à AWS Lake Formation du Guide du développeur AWS Lake Formation .

Considérations et restrictions

- Les métadonnées des colonnes sont visibles par les utilisateurs non autorisés dans certaines circonstances avec Avro et Custom SerDe
- Utilisation des autorisations Lake Formation pour les vues
- Contrôle d'accès précis de Lake Formation et groupes de travail Athena
- Emplacement des résultats de la requête Athena dans Simple Storage Service (Amazon S3) non enregistré dans Lake Formation
- Utilisation des groupes de travail Athena pour limiter l'accès à l'historique des requêtes
- Accès inter-comptes au catalogue de données
- Emplacements Amazon S3 chiffrés par CSE-KMS enregistrés auprès de Lake Formation
- Les emplacements des données enregistrées dans Lake Formation doivent se trouver dans des sous-répertoires de table
- Les requêtes de type CTAS (Create Table As Select) nécessitent des autorisations d'écriture sur Simple Storage Service (Amazon S3)
- L'autorisation DESCRIBE est requise pour la base de données par défaut

Les métadonnées des colonnes sont visibles par les utilisateurs non autorisés dans certaines circonstances avec Avro et Custom SerDe

L'autorisation au niveau des colonnes de Lake Formation empêche les utilisateurs d'accéder aux données des colonnes pour lesquelles ils ne disposent pas d'autorisations Lake Formation. Toutefois, dans certaines situations, les utilisateurs peuvent accéder aux métadonnées décrivant toutes les colonnes de la table, y compris les colonnes pour lesquelles ils ne disposent pas d'autorisations pour les données.

Cela se produit lorsque les métadonnées des colonnes sont stockées dans les propriétés des tables en utilisant le format de stockage Apache Avro ou en utilisant un sérialiseur/désérialiseur (SerDe) personnalisé dans lequel le schéma de table est défini dans les propriétés de table avec la définition. SerDe Lorsque vous utilisez Athena avec Lake Formation, nous vous recommandons d'examiner le contenu des propriétés des tables que vous enregistrez dans Lake Formation et, dans la mesure du possible, de limiter les informations stockées dans les propriétés des tables pour éviter que les métadonnées sensibles ne soient visibles par les utilisateurs.

Utilisation des autorisations Lake Formation pour les vues

Pour les données enregistrées dans Lake Formation, un utilisateur Athena ne peut créer une `VIEW` que s'il dispose des autorisations Lake Formation pour les tables, les colonnes et les emplacements

de données Simple Storage Service (Amazon S3) sources sur lesquels la `VIEW` est basée. Une fois qu'une `VIEW` est créée dans Athena, les autorisations Lake Formation peuvent être appliquées à la `VIEW`. Les autorisations au niveau des colonnes ne sont pas disponibles pour un `VIEW`. Les utilisateurs qui disposent d'autorisations Lake Formation pour une `VIEW` mais qui ne disposent pas d'autorisations pour la table et les colonnes sur lesquelles la vue est basée ne peuvent pas utiliser la `VIEW` pour interroger les données. Cependant, les utilisateurs disposant de cette combinaison d'autorisations peuvent utiliser des instructions telles que `DESCRIBE VIEW`, `SHOW CREATE VIEW` et `SHOW COLUMNS` pour afficher les métadonnées `VIEW`. Pour cette raison, veillez à aligner les autorisations de Lake Formation pour chaque `VIEW` sur les autorisations de la table sous-jacente. Les filtres de cellules définis sur une table ne s'appliquent pas à une `VIEW` pour cette table. Les noms des liens de ressource doivent avoir le même nom que la ressource dans le compte d'origine. Des limites supplémentaires s'appliquent à l'utilisation de vues dans une configuration intrecomptes. Pour de plus amples informations sur la configuration des autorisations pour les vues partagées entre comptes, consultez Accès inter-comptes au catalogue de données.

Contrôle d'accès précis de Lake Formation et groupes de travail Athena

Les utilisateurs du même groupe de travail Athena peuvent voir les données que le contrôle d'accès précis de Lake Formation a configurées pour être accessibles au groupe de travail. Pour plus d'informations sur l'utilisation du contrôle d'accès précis dans Lake Formation, voir Gérer le contrôle d'accès précis à l'aide de AWS Lake Formation sur le blog AWS Big Data.

Emplacement des résultats de la requête Athena dans Simple Storage Service (Amazon S3) non enregistré dans Lake Formation

Les emplacements des résultats des requêtes dans Simple Storage Service (Amazon S3) pour Athena ne peuvent pas être enregistrés dans Lake Formation. Les autorisations de Lake Formation ne limitent pas l'accès à ces emplacements. À moins que vous ne limitiez l'accès, les utilisateurs d'Athena peuvent accéder aux fichiers de résultats de requêtes et aux métadonnées alors qu'ils ne disposent pas d'autorisations Lake Formation pour les données. Pour éviter cela, nous vous recommandons d'utiliser des groupes de travail pour spécifier l'emplacement des résultats des requêtes et d'aligner l'appartenance à un groupe de travail sur les autorisations de Lake Formation. Vous pouvez ensuite utiliser les politiques d'autorisation IAM pour limiter l'accès aux emplacements des résultats des requêtes. Pour plus d'informations sur les résultats des requêtes, voir Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie.

Utilisation des groupes de travail Athena pour limiter l'accès à l'historique des requêtes

L'historique des requêtes d'Athena présente une liste des requêtes enregistrées et des chaînes de requête complètes. À moins que vous n'utilisiez des groupes de travail pour séparer l'accès aux historiques de requêtes, les utilisateurs d'Athena qui ne sont pas autorisés à interroger les données dans Lake Formation peuvent visualiser les chaînes de requêtes exécutées sur ces données, y compris les noms de colonnes, les critères de sélection, etc. Nous vous recommandons d'utiliser des groupes de travail pour séparer les historiques de requêtes, et d'aligner l'appartenance à un groupe de travail Athena sur les autorisations de Lake Formation pour en limiter l'accès. Pour plus d'informations, consultez Utilisation des groupes de travail pour contrôler l'accès aux requêtes et les coûts.

Accès inter-comptes au catalogue de données

Pour accéder à un catalogue de données dans un autre compte, vous pouvez utiliser la fonction AWS Glue inter-comptes d'Athena ou configurer l'accès inter-comptes dans Lake Formation.

Accès au catalogue de données Athena inter-comptes

Vous pouvez utiliser la fonction de AWS Glue catalogue multi-comptes d'Athena pour enregistrer le catalogue dans votre compte. Cette fonctionnalité n'est disponible que dans la version 2 du moteur Athena et les versions ultérieures, et est limitée à l'utilisation de la même région entre les comptes. Pour plus d'informations, consultez Enregistrer un compte AWS Glue Data Catalog depuis un autre compte.

Si une politique de ressources est configurée dans le catalogue de données à partager AWS Glue, elle doit être mise à jour pour autoriser l'accès au catalogue de données du compte B AWS Resource Access Manager et autoriser celui-ci à utiliser le catalogue de données du compte A, comme dans l'exemple suivant.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "ram.amazonaws.com"
        },
        "Action": "glue:ShareResource",
        "Resource": [
          "arn:aws:glue:<REGION>:<ACCOUNT-A>:table/*/*",
          "arn:aws:glue:<REGION>:<ACCOUNT-A>:database/*",
```

```
            "arn:aws:glue:<REGION>:<ACCOUNT-A>:catalog"
        ]
    },
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::<ACCOUNT-B>:root"
        },
        "Action": "glue:*",
        "Resource": [
            "arn:aws:glue:<REGION>:<ACCOUNT-A>:table/*/*",
            "arn:aws:glue:<REGION>:<ACCOUNT-A>:database/*",
            "arn:aws:glue:<REGION>:<ACCOUNT-A>:catalog"
        ]
    }
    ]
}
```

Pour plus d'informations, consultez Accès entre comptes aux catalogues de données AWS Glue.

Configuration de l'accès inter-comptes dans Lake Formation

AWS Lake Formation vous permet d'utiliser un seul compte pour gérer un catalogue de données central. Vous pouvez utiliser cette fonction pour implémenter l'accès entre comptes aux métadonnées du catalogue de données et aux données sous-jacentes. Par exemple, un compte propriétaire peut accorder à un autre compte (destinataire) une autorisation `SELECT` sur une table.

Pour qu'une base de données ou une table partagée apparaisse dans l'éditeur de requête Athena, vous devez créer un lien de ressource dans Lake Formation vers la base de données ou la table partagée. Lorsque le compte du destinataire dans Lake Formation interroge la table du propriétaire, il CloudTrailajoute l'événement d'accès aux données aux journaux du compte du destinataire et du compte du propriétaire.

Pour les vues partagées, gardez à l'esprit les points suivants :

- Les requêtes sont exécutées sur des liens de ressources cibles, et non sur la table ou la vue source, puis la sortie est partagée avec le compte cible.
- Il ne suffit pas de partager uniquement le point de vue. Toutes les tables impliquées dans la création de la vue doivent faire partie du partage entre comptes.
- Le nom du lien de la ressource créé sur les ressources partagées doit correspondre au nom de la ressource dans le compte du propriétaire. Si le nom ne correspond pas, un message d'erreur

tel que Failed analysis stored view 'awsdatacatalog 's'affiche. *my-lf-resource-link*. *my-lf-view*': ligne 3:3 : Le schéma *schema_name n'existe* pas apparaît.

Pour de plus amples informations sur l'accès inter-comptes dans Lake Formation, consultez les ressources suivantes dans le Guide du développeur AWS Lake Formation :

Permettre l'accès entre comptes

Mode de fonctionnement des liens des ressources dans Lake Formation

Journalisation entre comptes CloudTrail

Emplacements Amazon S3 chiffrés par CSE-KMS enregistrés auprès de Lake Formation

Les tables OTF (Open Table Format) telles qu'Apache Iceberg qui présentent les caractéristiques suivantes ne peuvent pas être interrogées avec Athena :

- Les tables sont basées sur les emplacements de données Amazon S3 enregistrés auprès de Lake Formation.
- Les objets d'Amazon S3 sont chiffrés à l'aide du chiffrement côté client (CSE).
- Le chiffrement utilise des clés AWS KMS gérées par le client ()`CSE_KMS`.

Pour interroger des tables non OTF (chiffrées à l'aide d'une `CSE_KMS` clé), ajoutez le bloc suivant à la politique de la AWS KMS clé que vous utilisez pour le chiffrement CSE. <KMS_KEY_ARN>est l'ARN de la AWS KMS clé qui chiffre les données. <IAM-ROLE-ARN>est l'ARN du rôle IAM qui enregistre la position Amazon S3 dans Lake Formation.

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "*"
    },
    "Action": "kms:Decrypt",
    "Resource": "<KMS-KEY-ARN>",
    "Condition": {
        "ArnLike": {
            "aws:PrincipalArn": "<IAM-ROLE-ARN>"
        }
    }
```

```
}
```

Les emplacements des données enregistrées dans Lake Formation doivent se trouver dans des sous-répertoires de table

Les tables partitionnées enregistrées dans Lake Formation doivent avoir des données partitionnées dans des répertoires qui sont des sous-répertoires de la table dans Simple Storage Service (Amazon S3). Par exemple, une table avec l'emplacement `s3://DOC-EXAMPLE-BUCKET/mytable` et les partitions `s3://DOC-EXAMPLE-BUCKET/mytable/dt=2019-07-11`, `s3://DOC-EXAMPLE-BUCKET/mytable/dt=2019-07-12`, etc. peut être enregistrée dans Lake Formation et interrogée à l'aide d'Athena. D'autre part, une table dont l'emplacement `s3://DOC-EXAMPLE-BUCKET/mytable` et les partitions se trouvent à `s3://DOC-EXAMPLE-BUCKET/dt=2019-07-11`, `s3://DOC-EXAMPLE-BUCKET/dt=2019-07-12`, etc., ne peut pas être enregistrée dans Lake Formation. Comme ces partitions ne sont pas des sous-répertoires de `s3://DOC-EXAMPLE-BUCKET/mytable`, elles ne peuvent pas non plus être lues par Athena.

Les requêtes de type CTAS (Create Table As Select) nécessitent des autorisations d'écriture sur Simple Storage Service (Amazon S3)

Les instructions CTAS (Create Table As Statements) nécessitent un accès en écriture à l'emplacement Simple Storage Service (Amazon S3) des tables. Pour exécuter des requêtes CTAS sur des données enregistrées dans Lake Formation, les utilisateurs d'Athena doivent disposer d'autorisations IAM leur permettant d'écrire dans la table des emplacements Simple Storage Service (Amazon S3), en plus des autorisations Lake Formation appropriées leur permettant de lire les emplacements des données. Pour plus d'informations, consultez Création d'une table à partir des résultats des requêtes (CTAS).

L'autorisation DESCRIBE est requise pour la base de données par défaut

L'autorisation DESCRIBE de Lake Formation est requise pour la base de données `default`. L'exemple de AWS CLI commande suivant accorde l'`DESCRIBE`autorisation d'accéder à la `default` base de données à l'utilisateur `datalake_user1` titulaire du AWS compte`111122223333`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "DESCRIBE" --resource '{ "Database": {"Name":"default"}}
```

Pour plus d'informations, consultez la rubrique Référence des autorisations de Lake Formation du Guide du développeur AWS Lake Formation .

Gestion des autorisations de Lake Formation et des utilisateurs d'Athena

Lake Formation fournit des informations d'identification pour interroger les magasins de données Simple Storage Service (Amazon S3) qui sont enregistrés dans Lake Formation. Si vous utilisiez auparavant des politiques IAM pour autoriser ou rejeter les autorisations de lecture des emplacements de données dans Simple Storage Service (Amazon S3), vous pouvez utiliser les autorisations Lake Formation à la place. Cependant, d'autres autorisations IAM sont toujours nécessaires.

Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Les sections suivantes résument les autorisations requises pour utiliser Athena afin d'interroger les données enregistrées dans Lake Formation. Pour plus d'informations, consultez la rubrique Sécurité dans AWS Lake Formation du Guide du développeur AWS Lake Formation .

Récapitulatif des autorisations

- Autorisations basées sur l'identité pour Lake Formation et Athena
- Autorisations Simple Storage Service (Amazon S3) pour les emplacements des résultats de requêtes Athena
- Appartenances des groupes de travail Athena dans l'historique des requêtes
- Autorisations Lake Formation aux données
- Autorisations IAM permettant d'écrire dans des emplacements Simple Storage Service (Amazon S3)
- Autorisations pour les données chiffrées, les métadonnées et les résultats de requête Athena
- Autorisations basées sur les ressources pour les compartiments Simple Storage Service (Amazon S3) dans des comptes externes (facultatif)

Autorisations basées sur l'identité pour Lake Formation et Athena

Toute personne utilisant Athena pour interroger des données enregistrées dans Lake Formation doit disposer d'une politique d'autorisations IAM qui autorise l'action `lakeformation:GetDataAccess`. AWS politique gérée : AmazonAthenaFullAccess autorise cette action. Si vous utilisez des politiques en ligne, veillez à mettre à jour les politiques d'autorisations afin d'autoriser cette action.

Dans Lake Formation, un administrateur de lac de données a l'autorisation de créer des objets de métadonnées tels que des bases de données et des tables, d'accorder des autorisations Lake

Formation à d'autres utilisateurs et d'enregistrer de nouveaux emplacements Simple Storage Service (Amazon S3). Pour enregistrer de nouveaux emplacements, des autorisations sont nécessaires pour le rôle lié au service de Lake Formation. Pour plus d'informations, consultez les rubriques Création d'un administrateur de lac de données et Autorisations de rôles liés à des services pour Lake Formation du Guide du développeur AWS Lake Formation .

Un utilisateur de Lake Formation peut utiliser Athena pour interroger des bases de données, des tables, des colonnes de table et des magasins de données Simple Storage Service (Amazon S3) sous-jacents, en fonction des autorisations Lake Formation qui lui ont été accordées par les administrateurs du lac de données. Les utilisateurs ne peuvent pas créer de bases de données ou de tables, ni enregistrer de nouveaux emplacements Simple Storage Service (Amazon S3) dans Lake Formation. Pour de plus amples informations, consultez Création d'un utilisateur de lac de données dans le Guide du développeur AWS Lake Formation .

Dans Athena, les politiques d'autorisation basées sur l'identité, y compris celles des groupes de travail Athena, contrôlent toujours l'accès aux actions Athena pour les utilisateurs de comptes Amazon Web Services. En outre, l'accès fédéré peut être fourni par l'authentification basée sur SAML disponible avec les pilotes Athena. Pour plus d'informations, consultez Utilisation des groupes de travail pour contrôler l'accès aux requêtes et les coûts, Politiques IAM pour l'accès aux groupes de travail et Activation de l'accès fédéré à l'API Athena.

Pour plus d'informations, consultez la rubrique Octroi d'autorisations Lake Formation du Guide du développeur AWS Lake Formation .

Autorisations Simple Storage Service (Amazon S3) pour les emplacements des résultats de requêtes Athena

Les emplacements des résultats des requêtes dans Simple Storage Service (Amazon S3) pour Athena ne peuvent pas être enregistrés dans Lake Formation. Les autorisations de Lake Formation ne limitent pas l'accès à ces emplacements. À moins que vous ne limitiez l'accès, les utilisateurs d'Athena peuvent accéder aux fichiers de résultats de requêtes et aux métadonnées alors qu'ils ne disposent pas d'autorisations Lake Formation pour les données. Pour éviter cela, nous vous recommandons d'utiliser des groupes de travail pour spécifier l'emplacement des résultats des requêtes et d'aligner l'appartenance à un groupe de travail sur les autorisations de Lake Formation. Vous pouvez ensuite utiliser les politiques d'autorisation IAM pour limiter l'accès aux emplacements des résultats des requêtes. Pour plus d'informations sur les résultats des requêtes, voir Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie.

Appartenances des groupes de travail Athena dans l'historique des requêtes

L'historique des requêtes d'Athena présente une liste des requêtes enregistrées et des chaînes de requête complètes. À moins que vous n'utilisiez des groupes de travail pour séparer l'accès aux historiques de requêtes, les utilisateurs d'Athena qui ne sont pas autorisés à interroger les données dans Lake Formation peuvent visualiser les chaînes de requêtes exécutées sur ces données, y compris les noms de colonnes, les critères de sélection, etc. Nous vous recommandons d'utiliser des groupes de travail pour séparer les historiques de requêtes, et d'aligner l'appartenance à un groupe de travail Athena sur les autorisations de Lake Formation pour en limiter l'accès. Pour plus d'informations, consultez Utilisation des groupes de travail pour contrôler l'accès aux requêtes et les coûts.

Autorisations Lake Formation aux données

Outre l'autorisation de base d'utiliser Lake Formation, les utilisateurs d'Athena doivent disposer d'autorisations Lake Formation pour accéder aux ressources qu'ils interrogent. Ces autorisations sont accordées et gérées par un administrateur Lake Formation. Pour de plus amples informations, consultez Sécurité et contrôle d'accès aux métadonnées et données dans le Guide du développeur AWS Lake Formation .

Autorisations IAM permettant d'écrire dans des emplacements Simple Storage Service (Amazon S3)

Les autorisations Lake Formation sur Simple Storage Service (Amazon S3) ne comprennent pas la possibilité d'écrire sur Simple Storage Service (Amazon S3). Les instructions CTAS (Create Table As Statements) nécessitent un accès en écriture à l'emplacement Simple Storage Service (Amazon S3) des tables. Pour exécuter des requêtes CTAS sur des données enregistrées dans Lake Formation, les utilisateurs d'Athena doivent disposer d'autorisations IAM leur permettant d'écrire dans la table des emplacements Simple Storage Service (Amazon S3), en plus des autorisations Lake Formation appropriées leur permettant de lire les emplacements des données. Pour plus d'informations, consultez Création d'une table à partir des résultats des requêtes (CTAS).

Autorisations pour les données chiffrées, les métadonnées et les résultats de requête Athena

Les données sources sous-jacentes dans Simple Storage Service (Amazon S3) et les métadonnées dans le catalogue de données qui sont enregistrées dans Lake Formation peuvent être chiffrées. Il n'y a aucun changement dans la manière dont Athena traite le chiffrement des résultats des requêtes lors de l'utilisation d'Athena pour interroger des données enregistrées dans Lake Formation. Pour plus d'informations, consultez Chiffrement des résultats des requêtes Athena stockées dans Simple Storage Service (Amazon S3).

- Chiffrement des données source : le chiffrement des données sources des emplacements de données Simple Storage Service (Amazon S3) est pris en charge. Les utilisateurs d'Athena qui interrogent des emplacements Simple Storage Service (Amazon S3) chiffrés enregistrés dans Lake Formation ont besoin d'autorisations pour chiffrer et déchiffrer les données. Pour plus d'informations sur les exigences, voir Options de chiffrement Simple Storage Service (Amazon S3) prises en charge et Autorisations pour les données chiffrées dans Simple Storage Service (Amazon S3).
- Chiffrement des métadonnées : le chiffrement des métadonnées dans le catalogue de données est pris en charge. Pour les principals qui utilisent Athena, les politiques basées sur l'identité doivent autoriser les actions `"kms:GenerateDataKey"`, `"kms:Decrypt"` et `"kms:Encrypt"` pour la clé utilisée pour chiffrer les métadonnées. Pour plus d'informations, consultez le Chiffrement du catalogue de données dans le Guide du développeur AWS Glue et Accès depuis Athena aux métadonnées cryptées dans le AWS Glue Data Catalog.

Autorisations basées sur les ressources pour les compartiments Simple Storage Service (Amazon S3) dans des comptes externes (facultatif)

Pour interroger un emplacement de données Simple Storage Service (Amazon S3) dans un compte différent, une politique IAM basée sur les ressources (politique de compartiment) doit autoriser l'accès à l'emplacement. Pour plus d'informations, consultez Accès inter-comptes aux compartiments Simple Storage Service (Amazon S3) dans Athena.

Pour plus d'informations sur l'accès à un catalogue de données dans un autre compte, voir Accès au catalogue de données Athena inter-comptes.

Application d'autorisations Lake Formation à des bases de données et tables existantes

Si vous utilisez Athena pour la première fois et que vous utilisez Lake Formation pour configurer l'accès aux données de requête, vous n'avez pas besoin de configurer des politiques IAM afin que les utilisateurs puissent lire les données Simple Storage Service (Amazon S3) et créer des métadonnées. Vous pouvez utiliser Lake Formation pour gérer les autorisations.

Il n'est pas nécessaire d'enregistrer les données dans Lake Formation et de mettre à jour les politiques d'autorisations IAM. Si les données ne sont pas enregistrées auprès de Lake Formation, les utilisateurs d'Athena disposant des autorisations appropriées dans Amazon S3 et AWS Glue, le cas échéant, peuvent continuer à interroger les données non enregistrées auprès de Lake Formation.

Si vous avez déjà des utilisateurs d'Athena qui interrogent des données non enregistrées auprès de Lake Formation, vous pouvez mettre à jour les autorisations IAM pour Amazon S3 (et le cas échéant) afin de pouvoir utiliser les AWS Glue Data Catalog autorisations de Lake Formation pour gérer l'accès des utilisateurs de manière centralisée. Pour obtenir l'autorisation de lire les emplacements de données Simple Storage Service (Amazon S3), vous pouvez mettre à jour les politiques basées sur les ressources et les politiques basées sur l'identité afin de modifier les autorisations Simple Storage Service (Amazon S3). Pour l'accès aux métadonnées, si vous avez configuré des politiques au niveau des ressources pour un contrôle d'accès précis AWS Glue, vous pouvez utiliser les autorisations de Lake Formation pour gérer l'accès à la place.

Pour plus d'informations, reportez-vous Accès détaillé aux bases de données et aux tables du AWS Glue Data Catalog à la section Mise à niveau des autorisations de AWS Glue données pour le AWS Lake Formation modèle dans le Guide du AWS Lake Formation développeur.

Utilisation de Lake Formation et des pilotes JDBC et ODBC d'Athena pour l'accès fédéré à Athena

Les pilotes JDBC et ODBC d'Athena prennent en charge la fédération basée sur SAML 2.0 avec Athena à l'aide de fournisseurs d'identité Okta et Microsoft Active Directory Federation Services (AD FS). En intégrant Amazon Athena à AWS Lake Formation, vous activez l'authentification basée sur SAML auprès d'Athena à l'aide des informations d'identification de l'entreprise. Avec Lake Formation et AWS Identity and Access Management (IAM), vous pouvez maintenir un contrôle d'accès précis au niveau des colonnes sur les données mises à la disposition de l'utilisateur SAML. Avec les pilotes JDBC et ODBC d'Athena, l'accès fédéré est disponible pour les outils ou les accès programmatiques.

Pour utiliser Athena afin d'accéder à une source de données contrôlée par Lake Formation, vous devez activer la fédération basée sur SAML 2.0 en configurant vos rôles de fournisseur d'identité (IdP) et (IAM). AWS Identity and Access Management Pour obtenir des instructions complètes, consultez Tutoriel : configuration de l'accès fédéré des utilisateurs d'Okta à Athena en utilisant Lake Formation et JDBC.

Prérequis

Pour utiliser Amazon Athena et Lake Formation pour l'accès fédéré, vous devez remplir les conditions suivantes :

- Gérer les identités de votre entreprise à l'aide d'un fournisseur d'identité existant basé sur SAML, tel que Okta ou Microsoft Active Directory Federation Services (AD FS).
- Vous l'utilisez AWS Glue Data Catalog comme magasin de métadonnées.

- Définir et gérer les autorisations dans Lake Formation pour accéder aux bases de données, tables et colonnes du AWS Glue Data Catalog. Pour plus d'informations, consultez le Guide du développeur AWS Lake Formation.
- Utiliser la version 2.0.14 ou une version ultérieure du pilote JDBC d'Athena ou la version 1.1.3 ou une version ultérieure du pilote ODBC d'Athena.

Considérations et restrictions

Lorsque vous utilisez le pilote JDBC ou ODBC d'Athena et Lake Formation pour configurer l'accès fédéré à Athena, gardez à l'esprit les points suivants :

- Actuellement, les pilotes JDBC et ODBC Athena prennent en charge les fournisseurs d'identité Okta, Microsoft Active Directory Federation Services (AD FS) et Azure AD. Bien que le pilote JDBC Athena possède une classe SAML générique qui peut être étendue pour utiliser d'autres fournisseurs d'identité, la prise en charge des extensions personnalisées permettant à d'autres fournisseurs d'identité () IdPs d'être utilisés avec Athena peut être limitée.
- L'accès fédéré à l'aide des pilotes JDBC et ODBC n'est pas compatible avec la fonctionnalité de propagation d'identité approuvée d'IAM Identity Center.
- Actuellement, vous ne pouvez pas utiliser la console Athena pour configurer la prise en charge de l'utilisation d'IdP et de SAML avec Athena. Pour configurer cette prise en charge, vous utilisez le fournisseur d'identité tiers, les consoles de gestion Lake Formation et IAM et le client de pilote JDBC ou ODBC.
- Vous devez comprendre la spécification SAML 2.0 et son fonctionnement avec votre fournisseur d'identité avant de configurer votre fournisseur d'identité et SAML pour une utilisation avec Lake Formation et Athena.
- Les fournisseurs SAML et les pilotes Athena JDBC et ODBC étant fournis par des tiers, l'assistance AWS en cas de problèmes liés à leur utilisation peut être limitée.

Rubriques

- Tutoriel : configuration de l'accès fédéré des utilisateurs d'Okta à Athena en utilisant Lake Formation et JDBC

Tutoriel : configuration de l'accès fédéré des utilisateurs d'Okta à Athena en utilisant Lake Formation et JDBC

Ce didacticiel explique comment configurer Okta AWS Lake Formation, les AWS Identity and Access Management autorisations et le pilote Athena JDBC pour permettre une utilisation fédérée d'Athena basée sur le protocole SAML. Lake Formation fournit à l'utilisateur basé sur SAML un contrôle d'accès précis aux données disponibles dans Athena. Pour configurer cette configuration, le didacticiel utilise la console de développement Okta, les consoles AWS IAM et Lake Formation, ainsi que l'outil SQL Workbench/J.

Prérequis

Ce tutoriel suppose que vous avez effectué les opérations suivantes :

- Création d'un compte Amazon Web Services. Pour créer un compte, visitez la page d'accueil d'Amazon Web Services.
- Configuration d'un emplacement de résultats de requête pour Athena dans Simple Storage Service (Amazon S3).
- Enregistrement d'un emplacement de compartiment de données Simple Storage Service (Amazon S3) dans Lake Formation.
- Défini une base de données et des tables dans le Catalogue de données AWS Glue qui pointent vers vos données dans Amazon S3.
  - Si vous n'avez pas encore défini de table, lancez un AWS Glue robot d'exploration ou utilisez Athena pour définir une base de données et une ou plusieurs tables pour les données auxquelles vous souhaitez accéder.
  - Ce tutoriel utilise une table basée sur le jeu de données des courses de taxi de NYC disponible dans le Registre des données ouvertes sur AWS. Le tutoriel utilise le nom de la base de données `tripdb` et le nom de la table `nyctaxi`.

Étapes du didacticiel

- Étape 1 : créer un compte Okta
- Étape 2 : Ajouter des utilisateurs et des groupes à Okta
- Étape 3 : configurer une application Okta pour l'authentification SAML
- Étape 4 : Création d'un fournisseur d'identité AWS SAML et d'un rôle IAM d'accès à Lake Formation
- Étape 5 : ajouter le rôle IAM et le fournisseur d'identité SAML à l'application Okta

Étape 1 : créer un compte Okta

Ce tutoriel utilise Okta comme fournisseur d'identité basé sur SAML. Si vous n'avez pas encore de compte Okta, vous pouvez en créer un gratuitement. Un compte Okta est nécessaire pour pouvoir créer une application Okta pour l'authentification SAML.

Création d'un compte

1. Pour utiliser Okta, rendez-vous sur la page d'inscription des développeurs Okta et créez un compte d'essai Okta gratuit. Le service Developer Edition est gratuit dans les limites spécifiées par Okta à developer.okta.com/pricing.
2. Lorsque vous recevez l'e-mail d'activation, activez votre compte.

   Un nom de domaine Okta vous sera attribué. Notez le nom de domaine pour référence. Plus tard, vous utiliserez le nom de domaine (*< okta-idp-domain >*) dans la chaîne JDBC qui se connecte à Athena.

Étape 2 : Ajouter des utilisateurs et des groupes à Okta

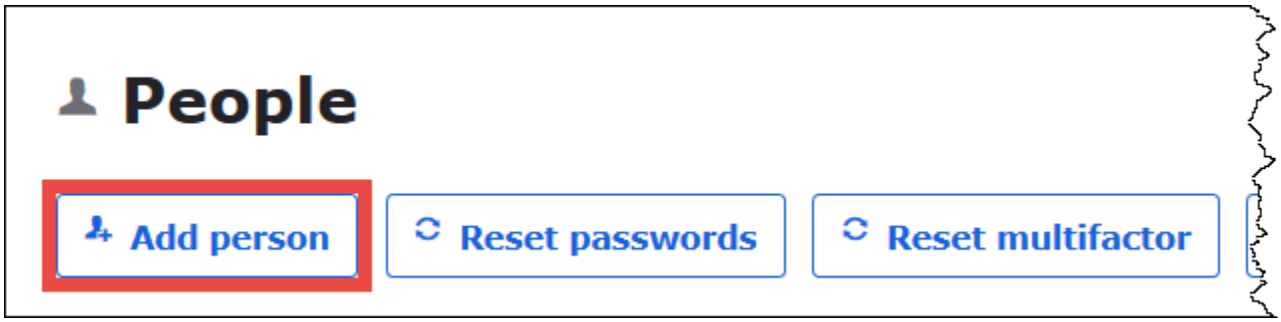
Dans cette étape, vous utilisez la console Okta pour effectuer les tâches suivantes :

- Créer deux utilisateurs Okta.
- Créer deux groupes Okta.
- Ajouter un utilisateur Okta à chaque groupe Okta.

Ajout d'utilisateurs à Okta

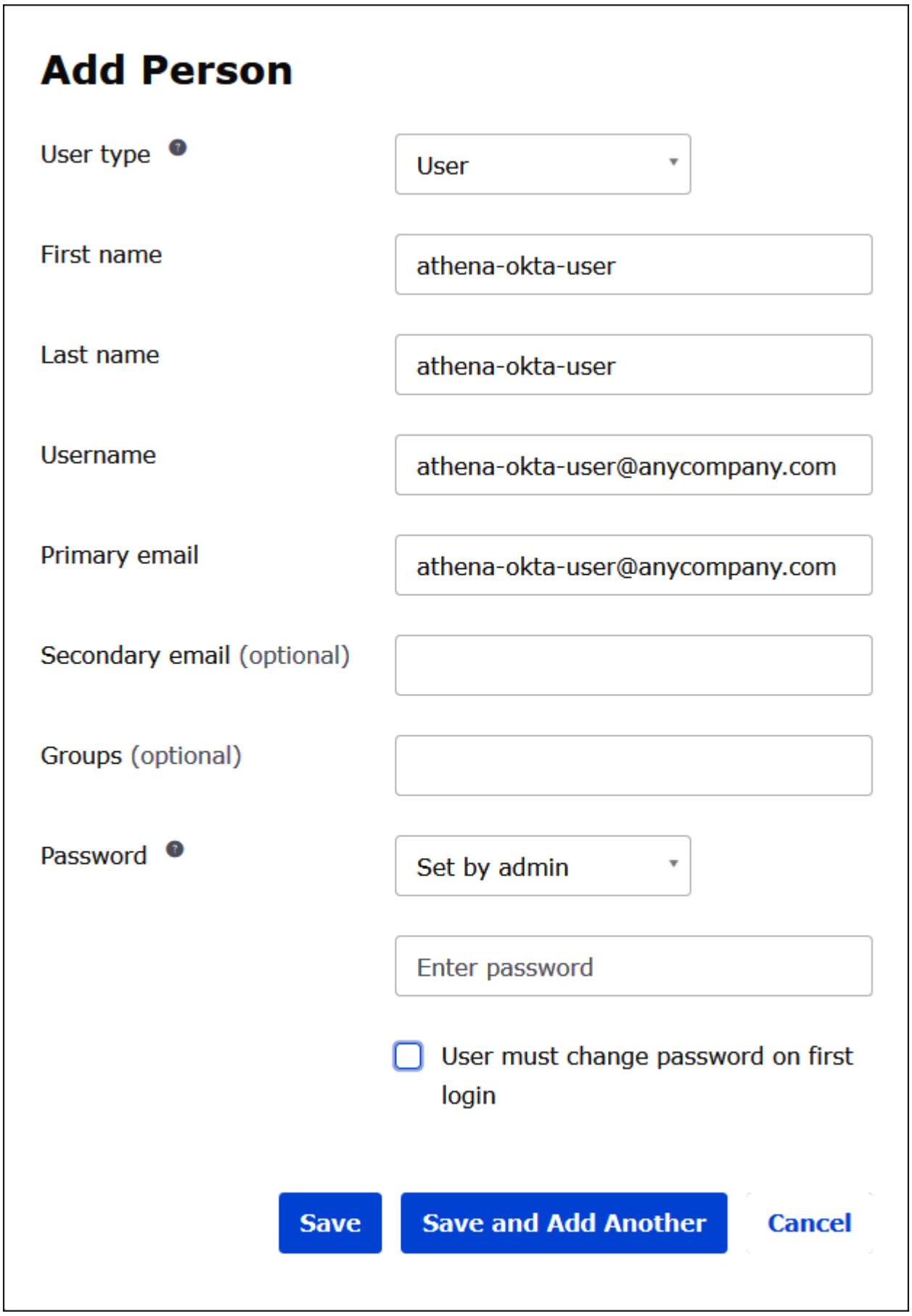
1. Après avoir activé votre compte Okta, connectez-vous en tant qu'utilisateur administratif au domaine Okta attribué.
2. Dans le panneau de navigation de gauche, choisissez Directory (Répertoire), puis choisissez People (Personnes).

3. Choisissez Add Person (Ajouter une personne) pour ajouter un nouvel utilisateur qui accédera à Athena via le pilote JDBC.

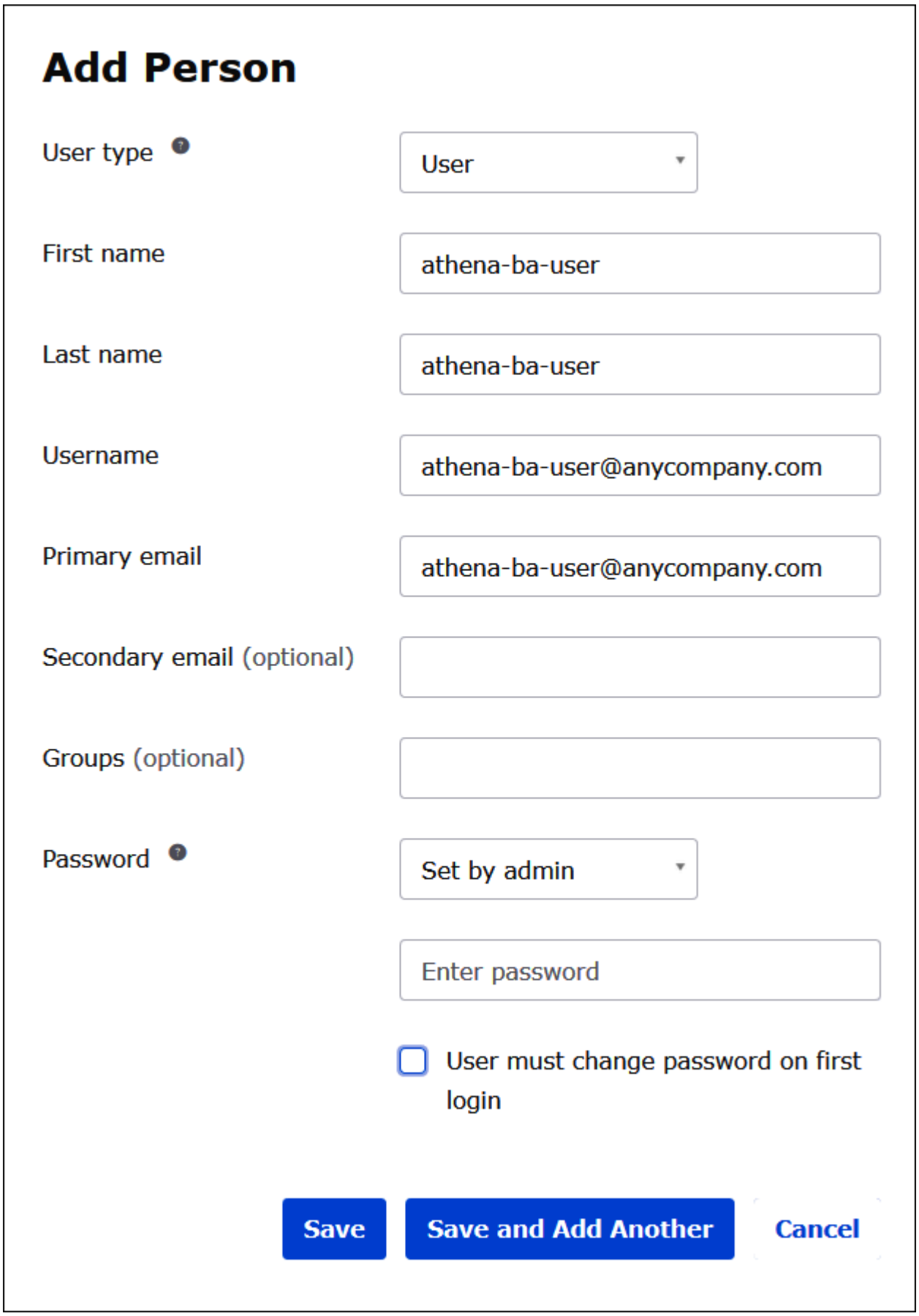


4. Dans la boîte de dialogue Add Person (Ajouter une personne), sasissez les informations requises.

   - Saisissez les valeurs pour First name (Prénom) et Last name (Nom de famille). Ce tutoriel utilise *athena-okta-user*.
   - Saisissez un Username (Nom d'utilisateur) et un Primary email (E-mail principal). Ce didacticiel utilise *athena-okta-user@anycompany .com*.
   - Pour Password (Mot de passe), choisissez Set by admin (Défini par l'administrateur), puis fournissez un mot de passe. Ce tutoriel désactive l'option User must change password on first login (L'utilisateur doit changer son mot de passe à la première connexion) ; vos exigences en matière de sécurité peuvent varier.


Add Person
User type
User
First name
athena-okta-user
Last name
athena-okta-user
Username
athena-okta-user@anycompany.com
Primary email
athena-okta-user@anycompany.com
Secondary email (optional)
Groups (optional)
Password
Set by admin
Enter password
User must change password on first login
Save
Save and Add Another
Cancel

5. Choisissez Save and Add Another (Enregistrer et ajouter un autre type).
6. Saisissez les informations pour un autre utilisateur. Cet exemple ajoute l'utilisateur d'analyse commerciale *athena-ba-user@anycompany .com*.

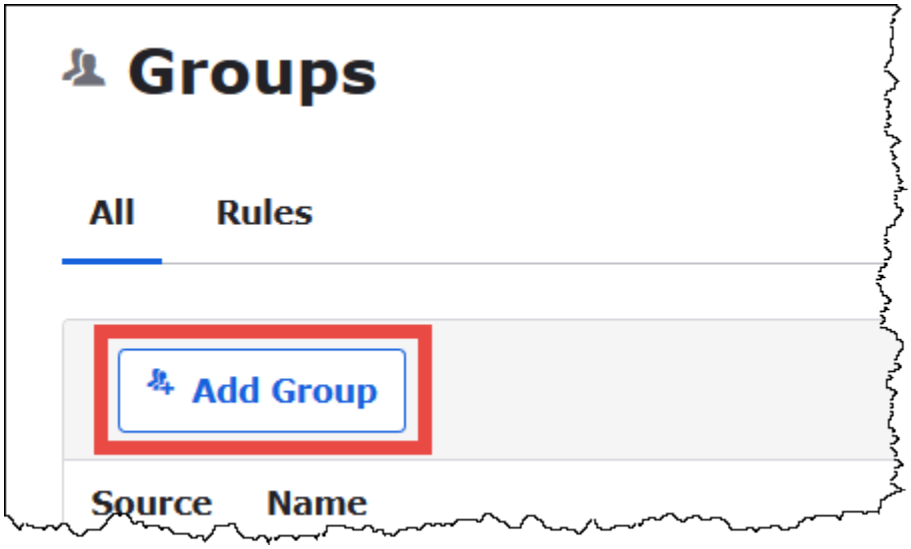

Add Person
User type
User
First name
athena-ba-user
Last name
athena-ba-user
Username
athena-ba-user@anycompany.com
Primary email
athena-ba-user@anycompany.com
Secondary email (optional)
Groups (optional)
Password
Set by admin
Enter password
User must change password on first login
Save
Save and Add Another
Cancel

7. Choisissez Enregistrer.

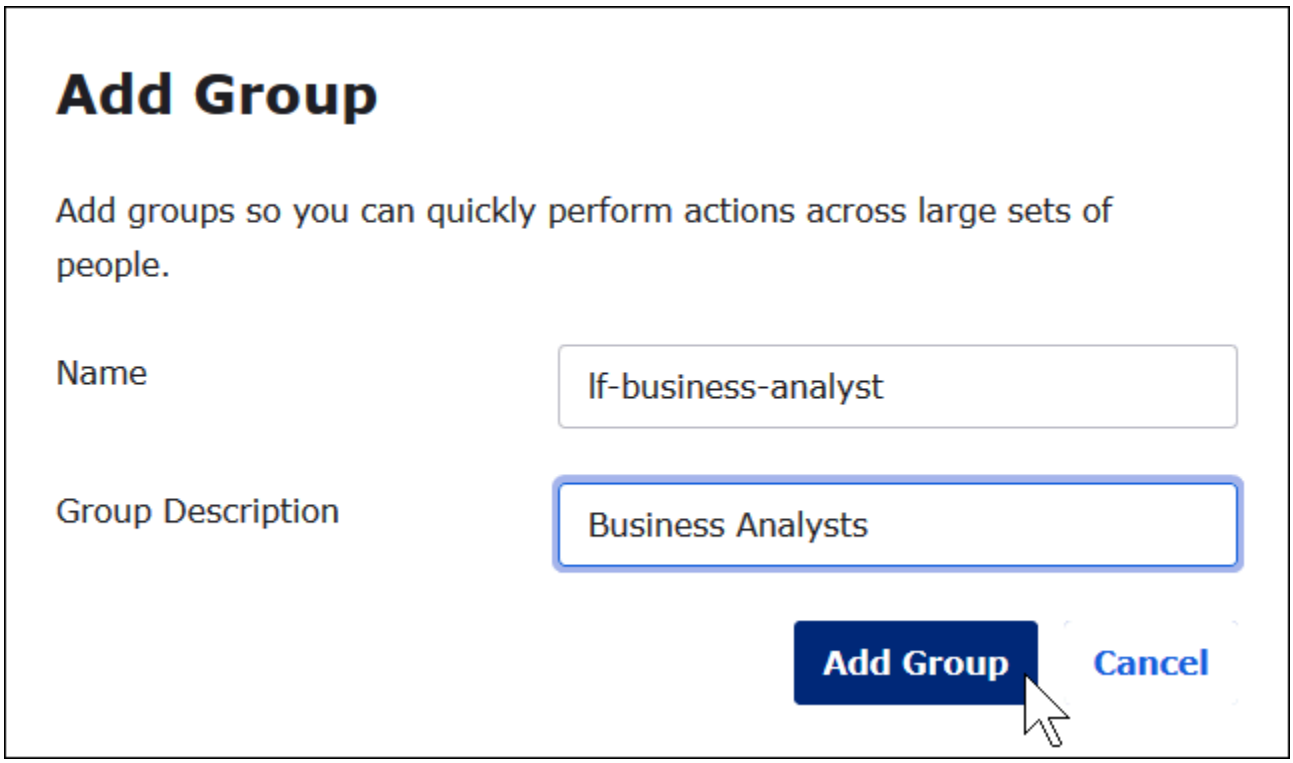
Dans la procédure suivante, vous fournissez l'accès à deux groupes Okta par le biais du pilote JDBC d'Athena en ajoutant un groupe « Analystes commerciaux » et un groupe « Développeurs ».

Ajout de groupes Okta

1. Dans le panneau de navigation de gauche, choisissez Directory (Répertoire), puis choisissez Groups (Groupes).
2. Sur la page Groups (Groupes), choisissez Add Group (Ajouter un groupe).



3. Dans la boîte de dialogue Add Group (Ajouter un groupe), saisissez les informations requises.
   - Pour Name (Nom), saisissez *lf-business-analyst*.
   - Pour Group Description (Description du groupe), saisissez *Business Analysts* (Analystes commerciaux).

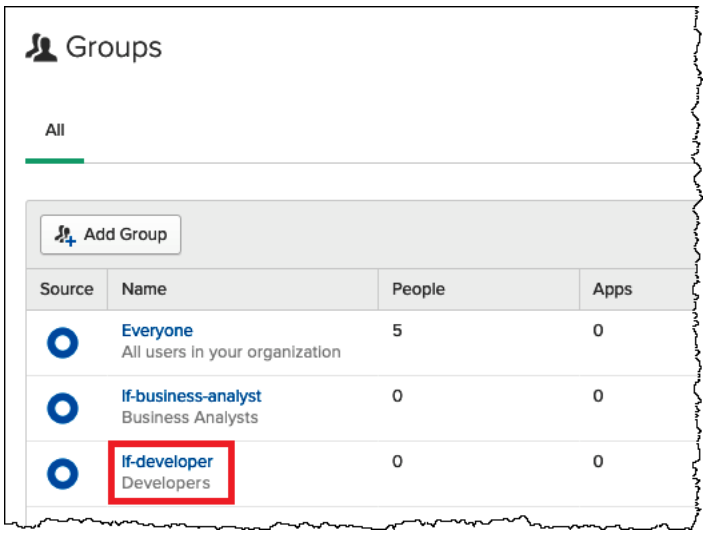


4. Choisissez Add Group (Ajouter un groupe).
5. Sur la page Groups (Groupes), choisissez à nouveau Add Group (Ajouter un groupe). Cette fois, vous allez saisir les informations pour le groupe des développeurs.
6. Saisissez les informations requises.
   - Pour Name (Nom), saisissez `lf-developer`.
   - Pour Group Description (Description du groupe), saisissez `Developers` (Développeurs).
7. Choisissez Add Group (Ajouter un groupe).

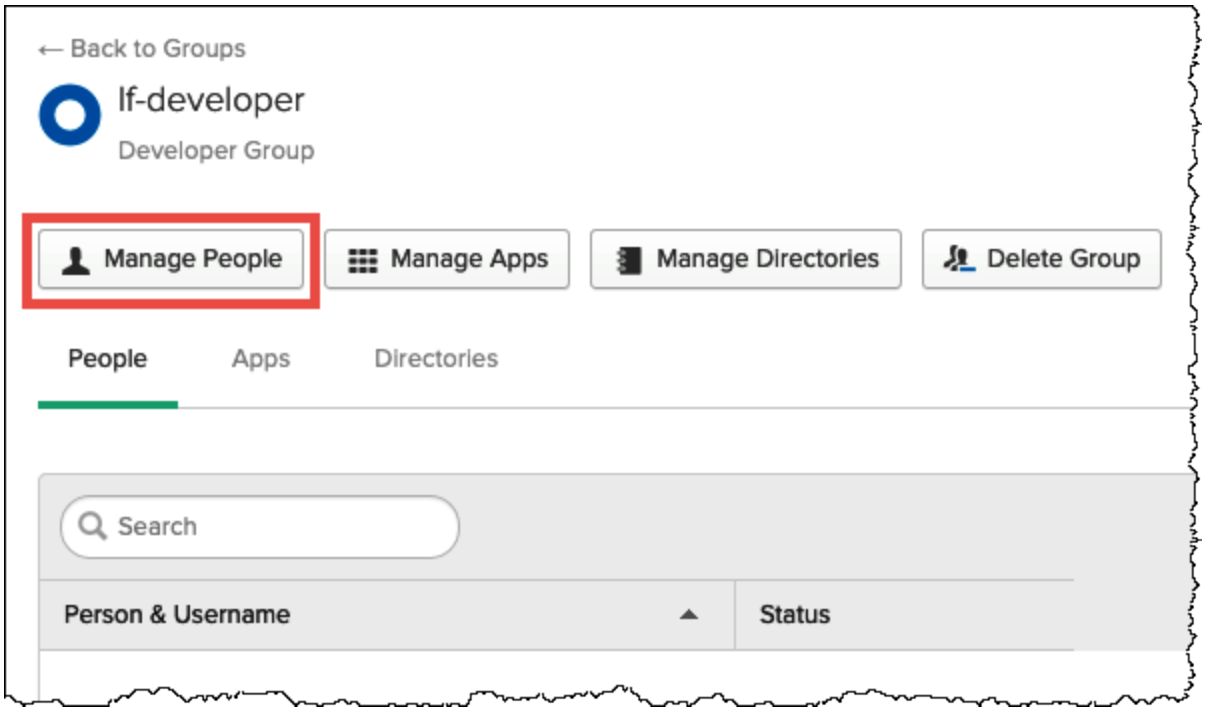
Maintenant que vous avez deux utilisateurs et deux groupes, vous pouvez ajouter un utilisateur à chaque groupe.

Ajout d'utilisateurs à un groupe

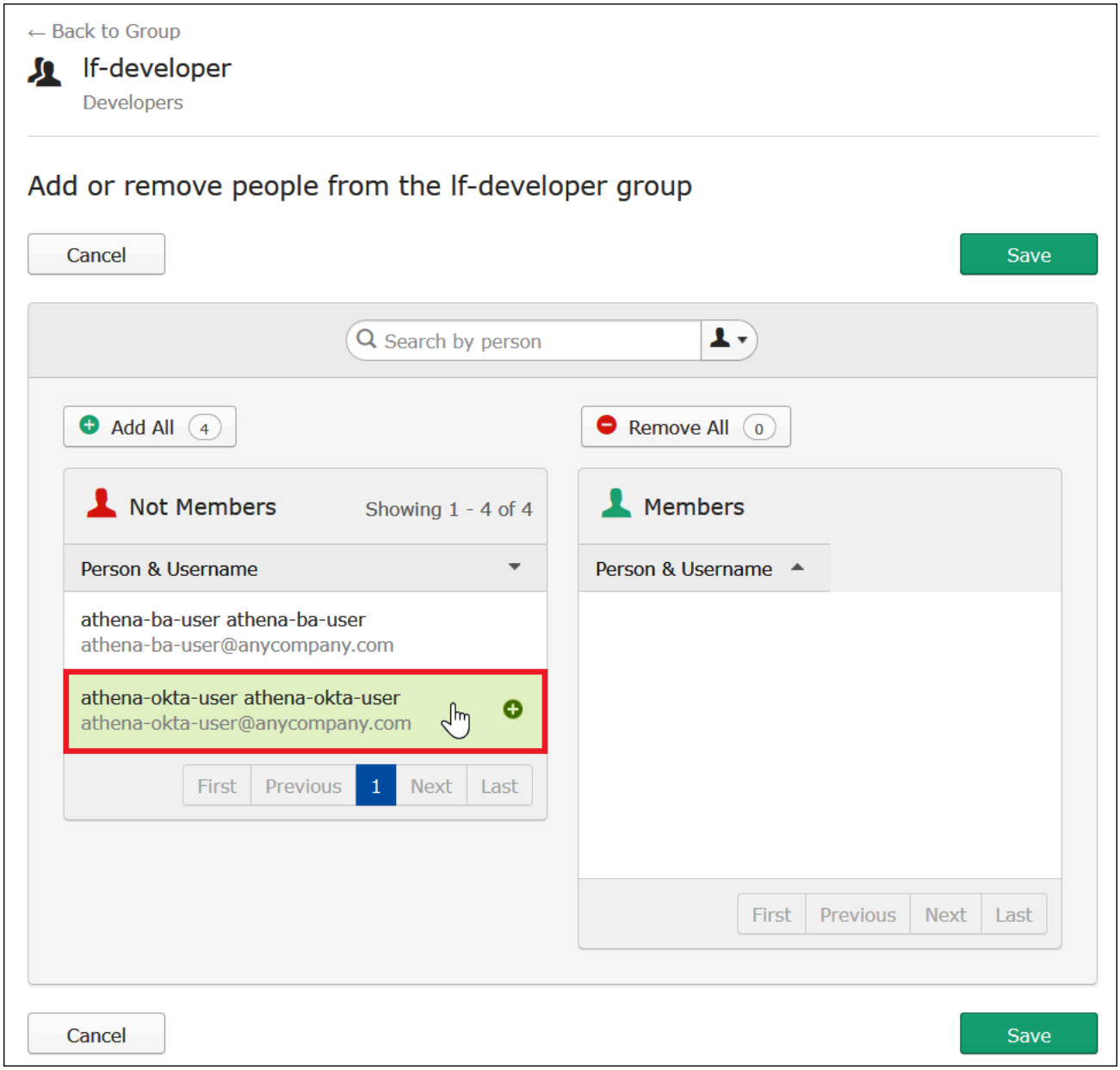
1. Sur la page Groups (Groupes), choisissez le groupe lf-developer que vous venez de créer. Vous allez ajouter à ce groupe l'un des utilisateurs Okta que vous avez créé en tant que développeur.



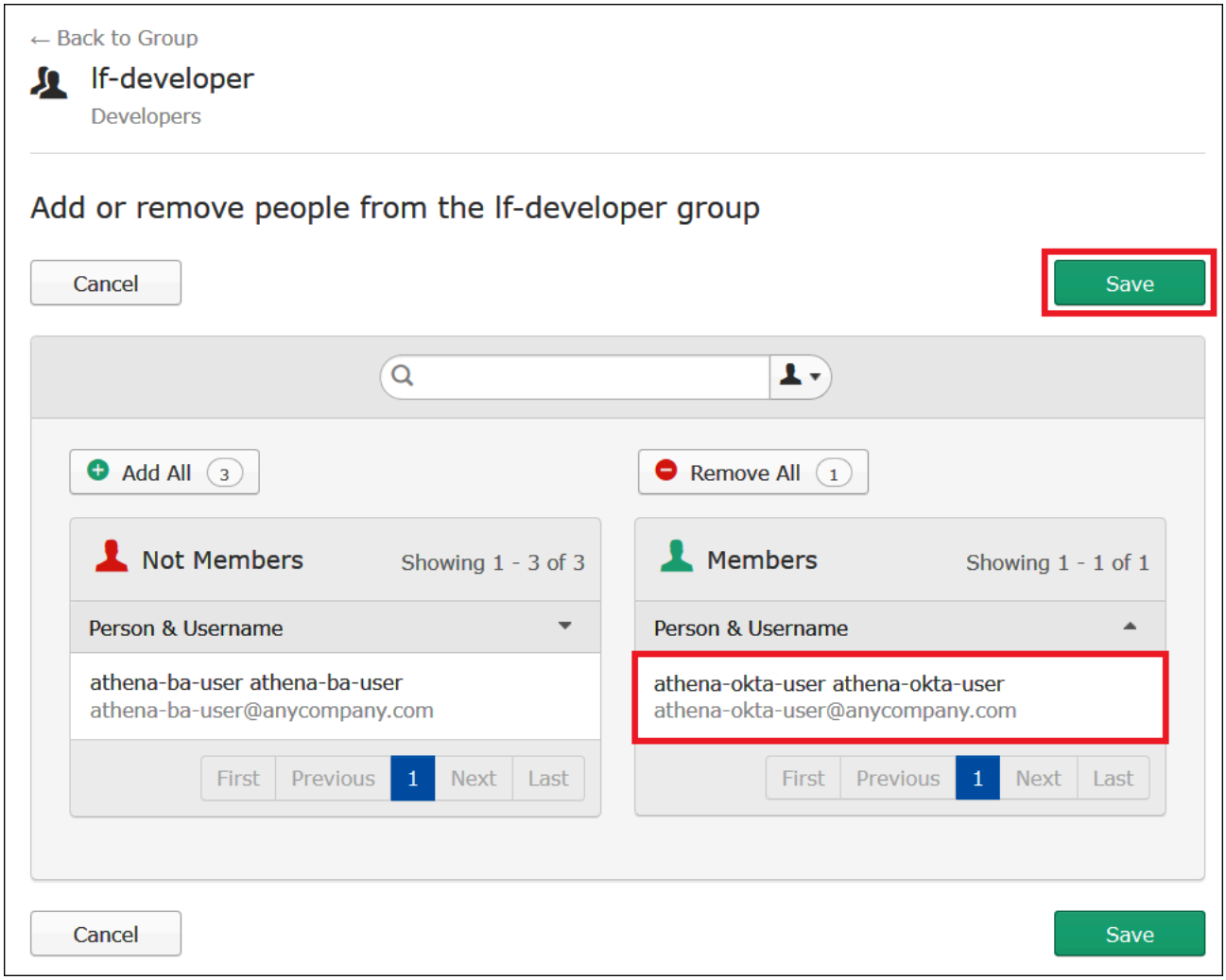
2. Choisissez Manage People (Gérer personnes).



3. Dans la liste des non-membres, sélectionnez athena-okta-user.

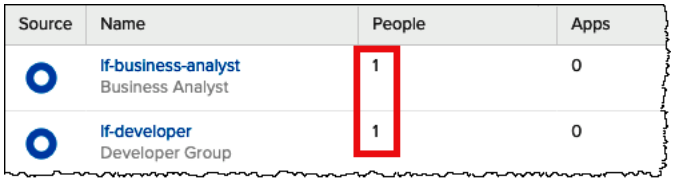


L'entrée de l'utilisateur passe de la liste Not Members (Non membres), à gauche, à la liste Members (Membres), à droite.



4. Choisissez Enregistrer.
5. Choisissez Back to Group (Retour au groupe), ou choisissez Directory (Répertoire), puis choisissez Groups (Groupes).
6. Choisissez le lf-business-analystgroupe.
7. Choisissez Manage People (Gérer personnes).
8. Ajoutez le athena-ba-userà la liste des membres du lf-business-analystgroupe, puis choisissez Enregistrer.
9. Choisissez Back to Group (Retour au groupe), ou choisissez Directory (Répertoire), Groups (Groupes).

   La page Groups (Groupes) montre maintenant que chaque groupe a un utilisateur Okta.



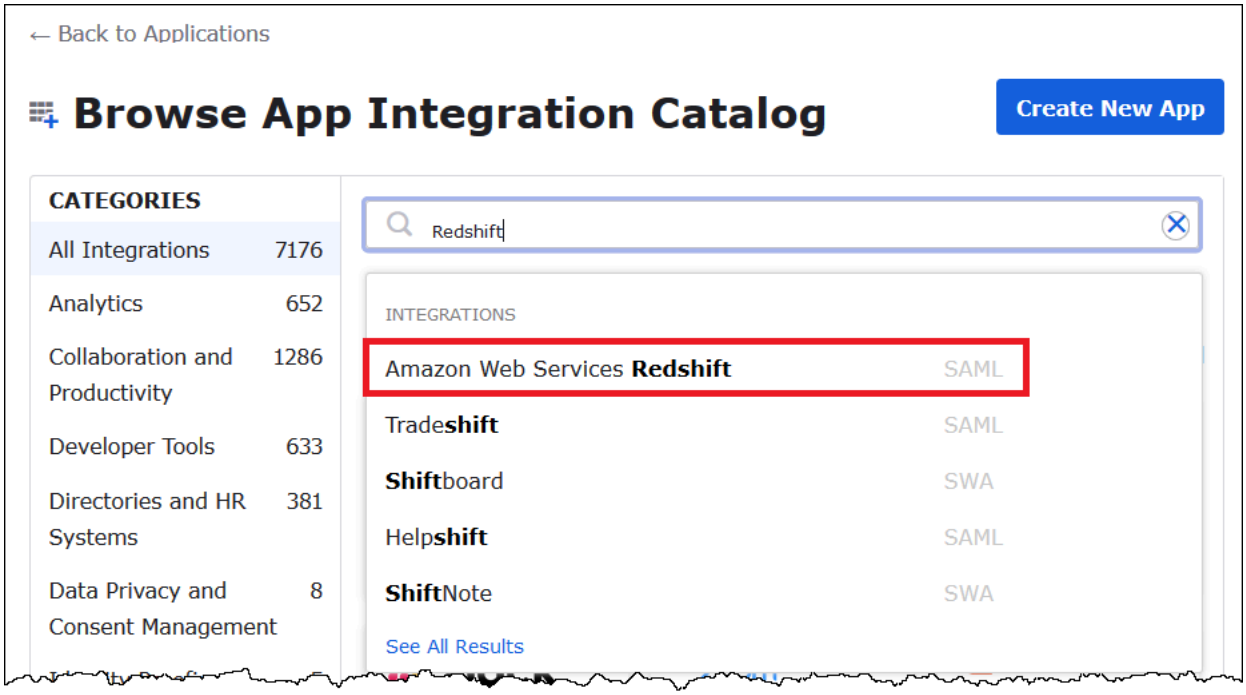
Étape 3 : configurer une application Okta pour l'authentification SAML

Dans cette étape, vous utilisez la console du développeur Okta pour effectuer les tâches suivantes :

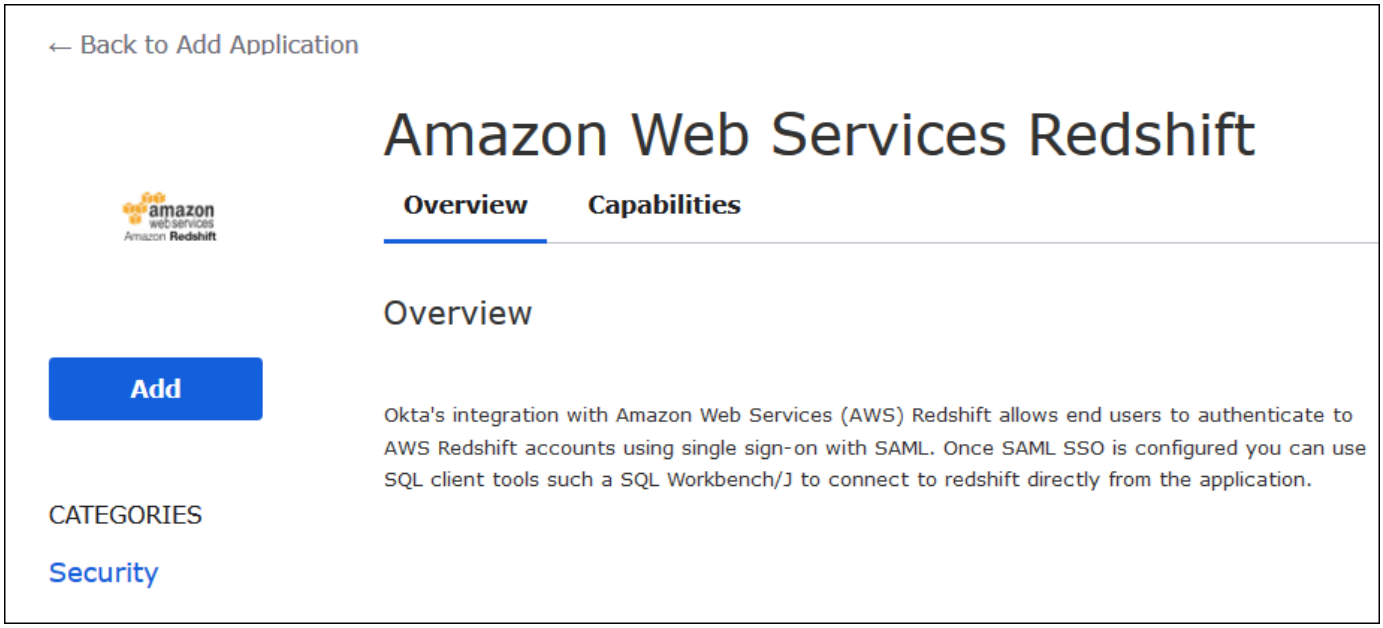
- Ajoutez une application SAML à utiliser avec AWS.
- Affecter l'application à l'utilisateur Okta.
- Affecter l'application à un groupe Okta.
- Téléchargez les métadonnées du fournisseur d'identité résultant pour une utilisation ultérieure avec AWS.

Ajout d'une application pour l'authentification SAML

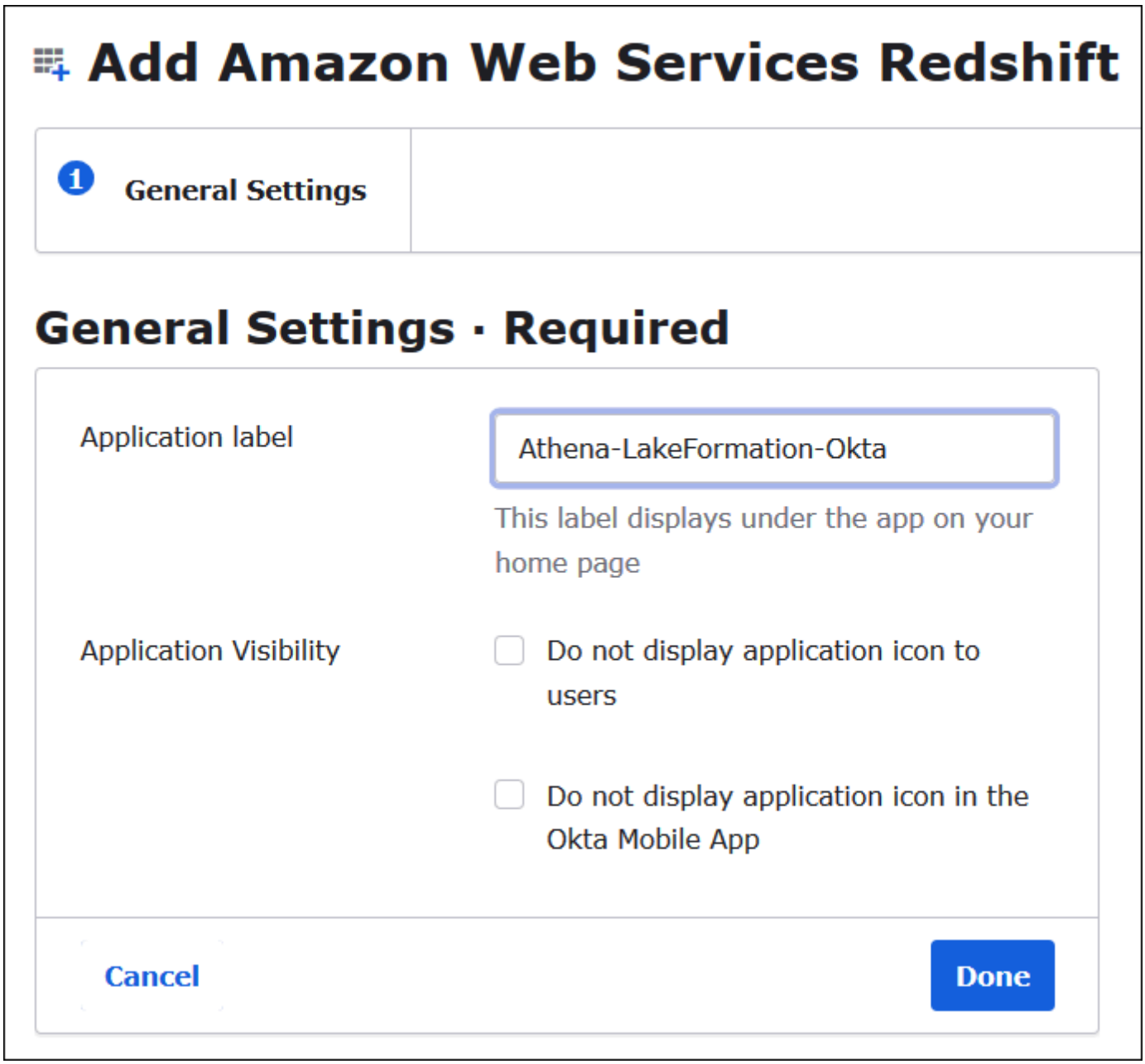
1. Dans le panneau de navigation Okta, choisissez Applications, Applications afin que vous puissiez configurer une application Okta pour l'authentification SAML à Athena.
2. Cliquez sur Browse App Catalog (Parcourir le catalogue d'applications).
3. Dans la zone de recherche, saisissez **`Redshift`**.
4. Choisissez Amazon Web Services Redshift. L'application Okta de ce tutoriel utilise l'intégration SAML existante pour Amazon Redshift.



5. Sur la page Amazon Web Services Redshift, choisissez Add (Ajouter) pour créer une application basée sur SAML pour Amazon Redshift.



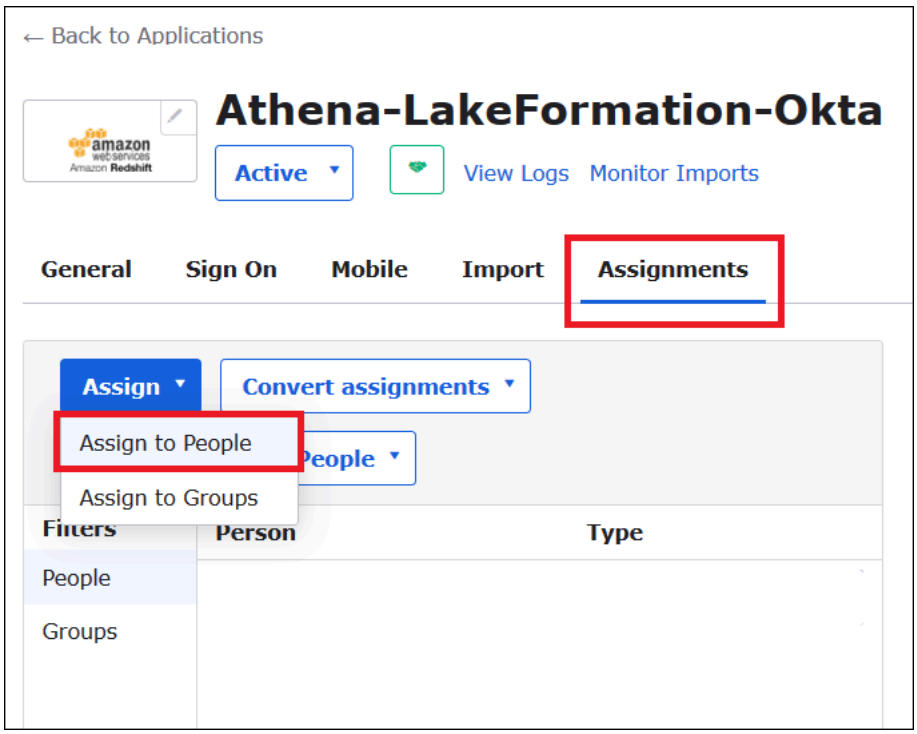
6. Pour Application label (Étiquette d'application), saisissez `Athena-LakeFormation-Okta`, puis Done (Terminé).



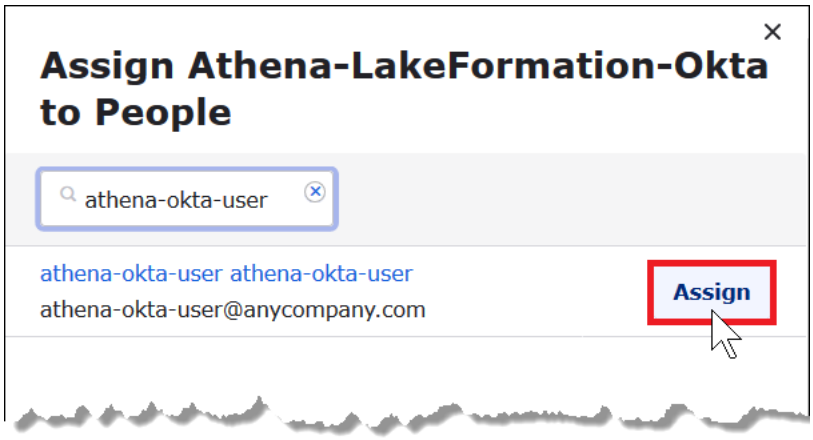
Maintenant que vous avez créé une application Okta, vous pouvez l'affecter aux utilisateurs et aux groupes que vous avez créés.

Affectation d'une application à des utilisateurs et des groupes

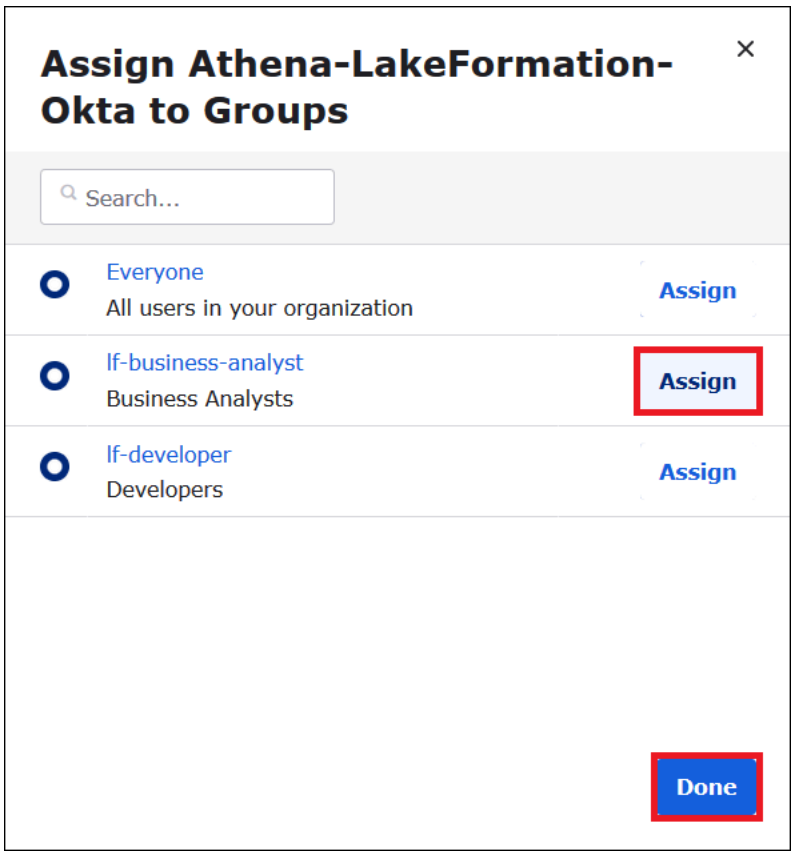
1. Sur la page Applications, choisissez l'application Athena- LakeFormation -Okta.
2. Dans l'onglet Assignments (Affectations), choisissez Assign (Affecter), Assign to People (Affecter à des personnes).



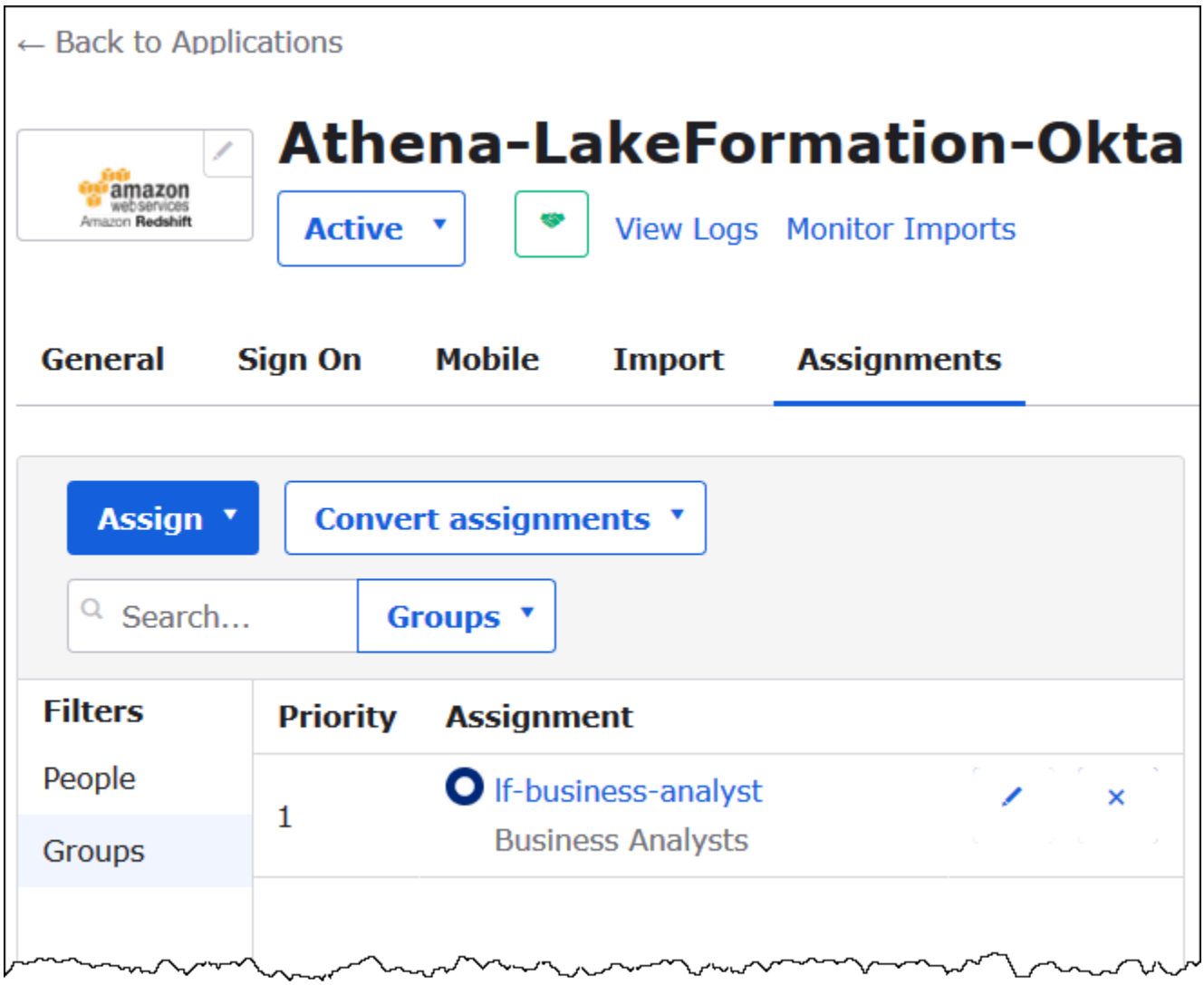
3. Dans la boîte de dialogue Attribuer Athena- LakeFormation -Okta à des personnes, recherchez l'athena-okta-userutilisateur que vous avez créé précédemment.
4. Choisissez Assign (Affecter) pour affecter l'utilisateur à l'application.



5. Choisissez Save and Go Back (Sauvegarder et revenir).
6. Sélectionnez Exécuté.
7. Dans l'onglet Attributions de l'application Athena- LakeFormation -Okta, choisissez Attribuer, Attribuer aux groupes.
8. Pour lf-business-analyst, choisissez Affecter pour attribuer l'application Athena- LakeFormation -Okta au lf-business-analystgroupe, puis cliquez sur Terminé.



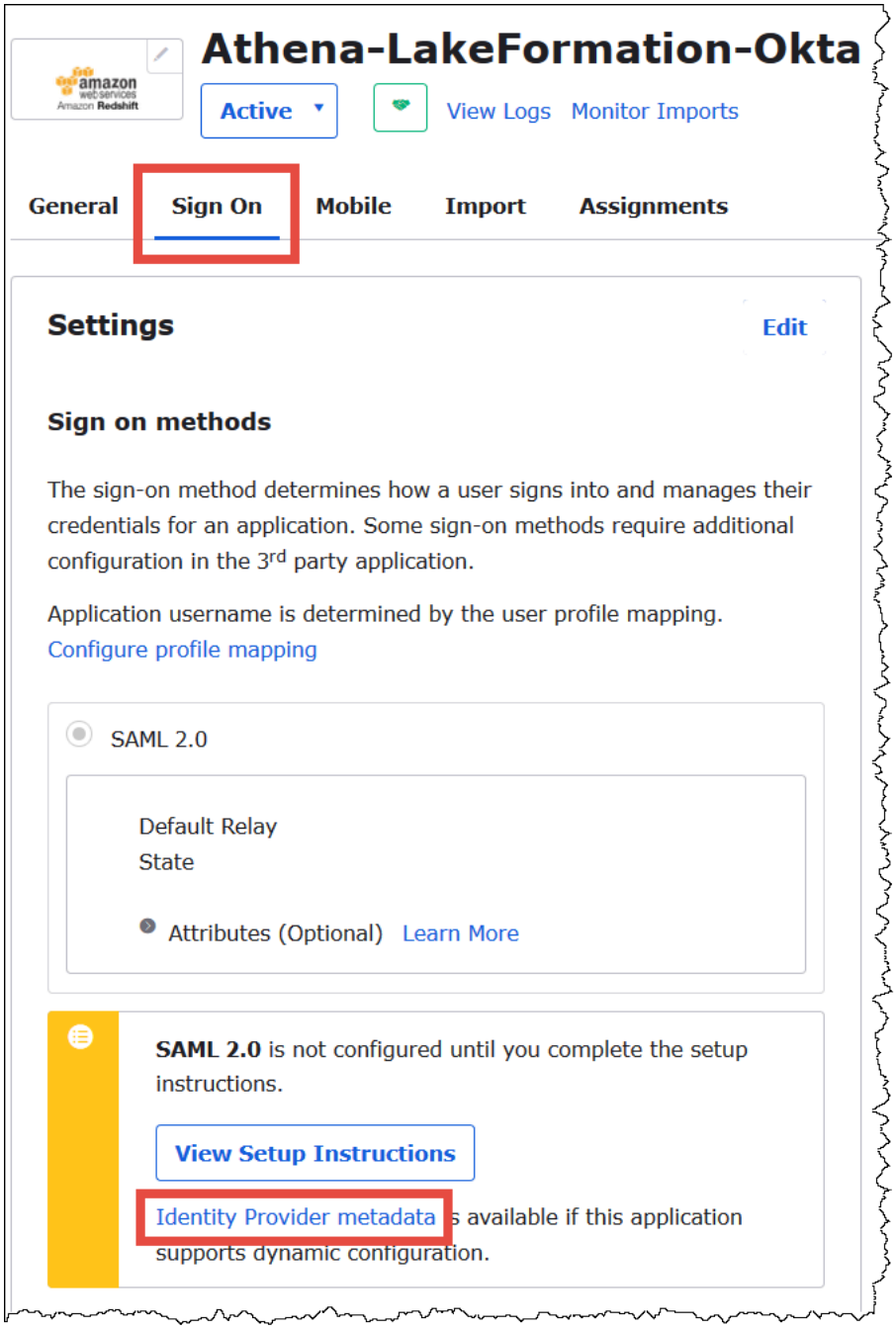
Le groupe s'affiche dans la liste des groupes pour l'application.



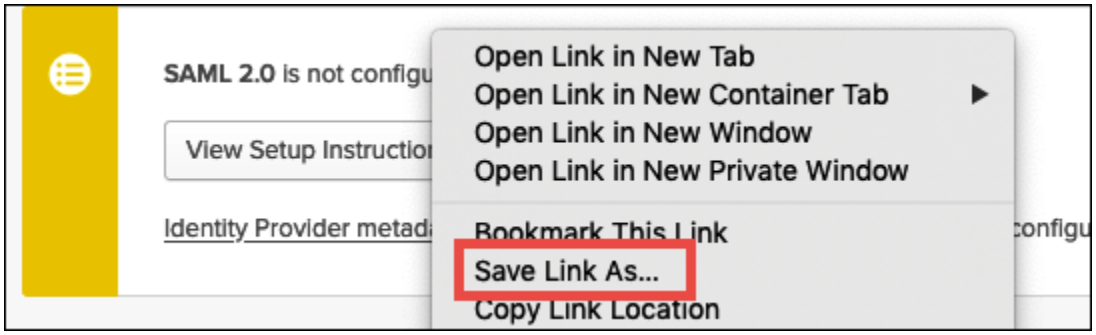
Vous pouvez maintenant télécharger les métadonnées de l'application du fournisseur d'identité à utiliser avec AWS.

Téléchargement des métadonnées de l'application

1. Choisissez l'onglet Sign On (Authentification) de l'application Okta, puis cliquez à droite sur Identity Provider metadata (Métadonnées du fournisseur d'identité).



2. Choisissez Save Link As (Enregistrer le lien sous) pour enregistrer les métadonnées du fournisseur d'identité, qui sont au format XML, dans un fichier. Donnez-lui un nom que vous reconnaissez (par exemple, `Athena-LakeFormation-idp-metadata.xml`).



Étape 4 : Création d'un fournisseur d'identité AWS SAML et d'un rôle IAM d'accès à Lake Formation

Au cours de cette étape, vous utilisez la console AWS Identity and Access Management (IAM) pour effectuer les tâches suivantes :

- Créer un fournisseur d'identité pour AWS.
- Créer un rôle IAM pour l'accès à Lake Formation.
- Ajoutez la politique AmazonAthenaFullAccess gérée au rôle.
- Ajoutez une politique pour Lake Formation et AWS Glue au rôle.
- Ajouter une politique pour les résultats des requêtes Athena au rôle.

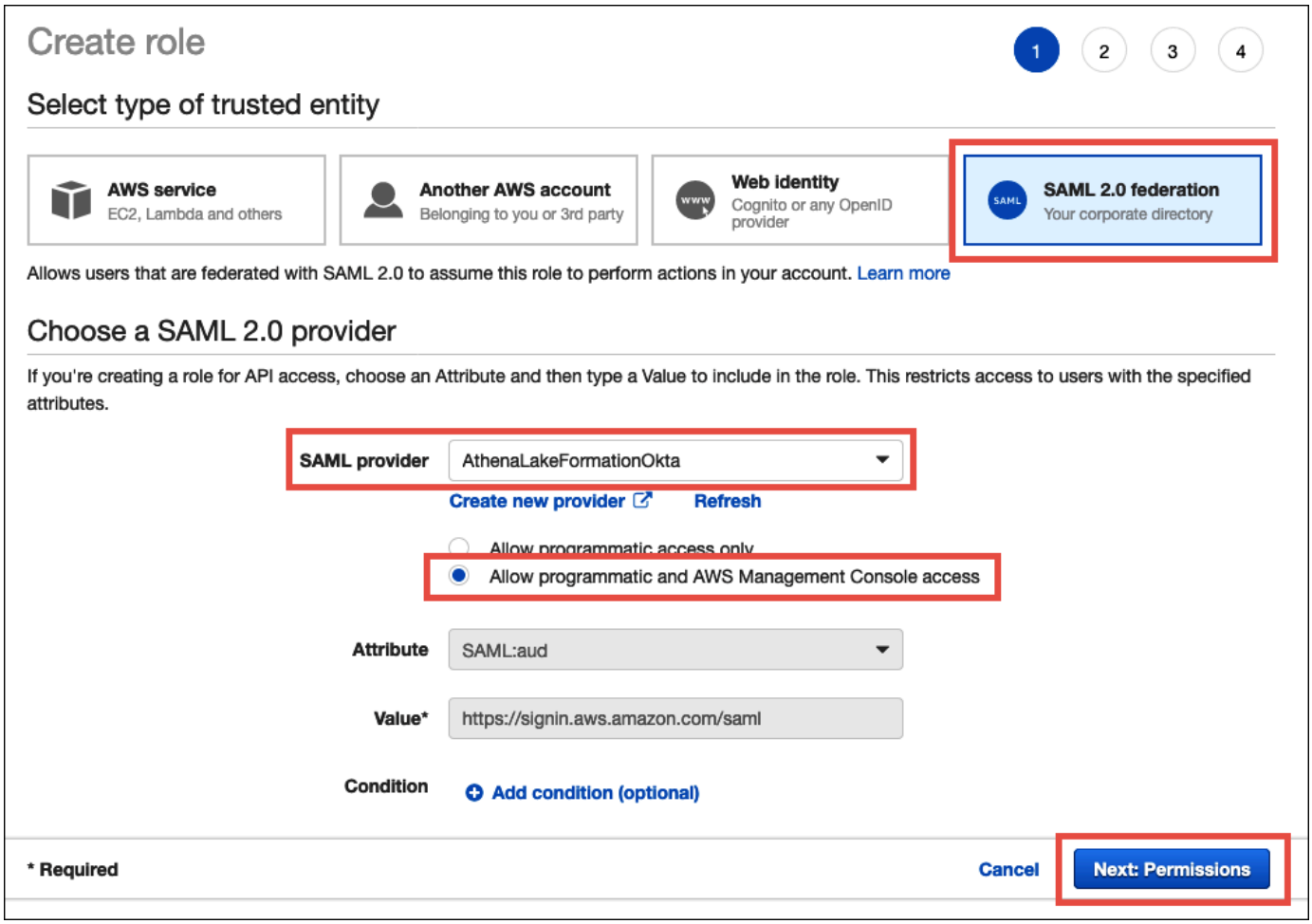
Pour créer un fournisseur d'identité AWS SAML

1. Connectez-vous à la console du Amazon Web Services account (compte Amazon Web Services) en tant qu'Amazon Web Services account administrator (administrateur de compte Amazon Web Services) et accédez à la console IAM (https://console.aws.amazon.com/iam/).
2. Dans le panneau de navigation, choisissez Identity providers (Fournisseurs d'identité), puis Add provider (Ajouter un fournisseur).
3. Sur l'écran Configure provider (Configurer le fournisseur), saisissez les informations suivantes :
   - Pour Provider type (Type de fournisseur), choisissez SAML.
   - Pour Provider name (Nom du fournisseur), saisissez `AthenaLakeFormationOkta`.
   - Pour Metadata document (Document de métadonnées), utilisez l'option Choose file (Choisir un fichier) pour téléverser le fichier XML de métadonnées du fournisseur d'identité (IdP) que vous avez téléchargé.
4. Choisissez Add provider (Ajouter un fournisseur).

Ensuite, vous créez un rôle IAM pour AWS Lake Formation y accéder. Vous ajoutez deux politiques en ligne au rôle. Une politique fournit des autorisations pour accéder à Lake Formation et aux AWS Glue API. L'autre politique donne accès à Athena et à l'emplacement des résultats des requêtes Athena dans Simple Storage Service (Amazon S3).

Pour créer un rôle IAM à des fins d'accès AWS Lake Formation

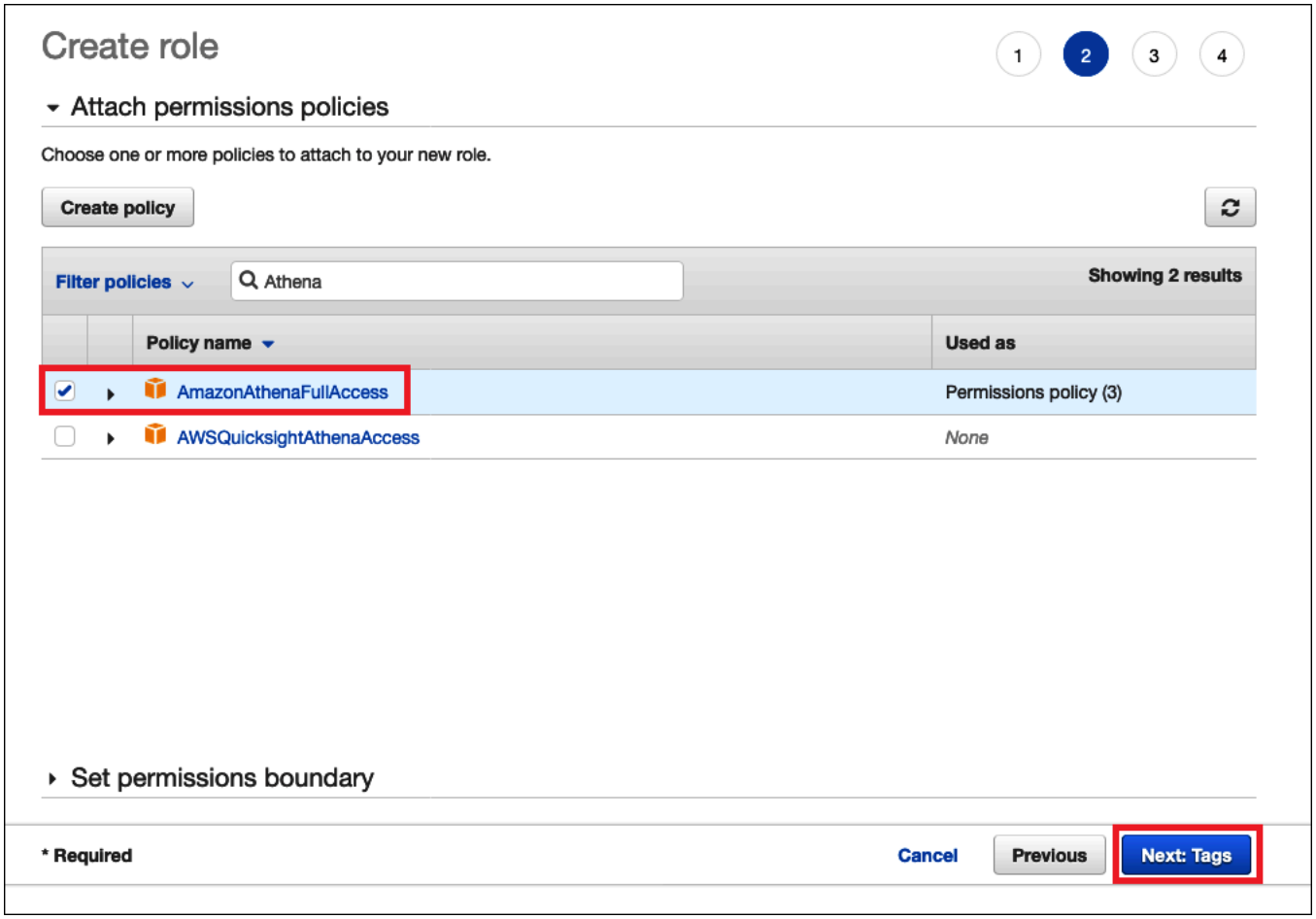
1. Dans le panneau de navigation de la console IAM, choisissez Roles (Rôles), puis Create role (Créer un rôle).
2. Sur la page Create role (Créer un rôle), suivez les étapes ci-dessous :



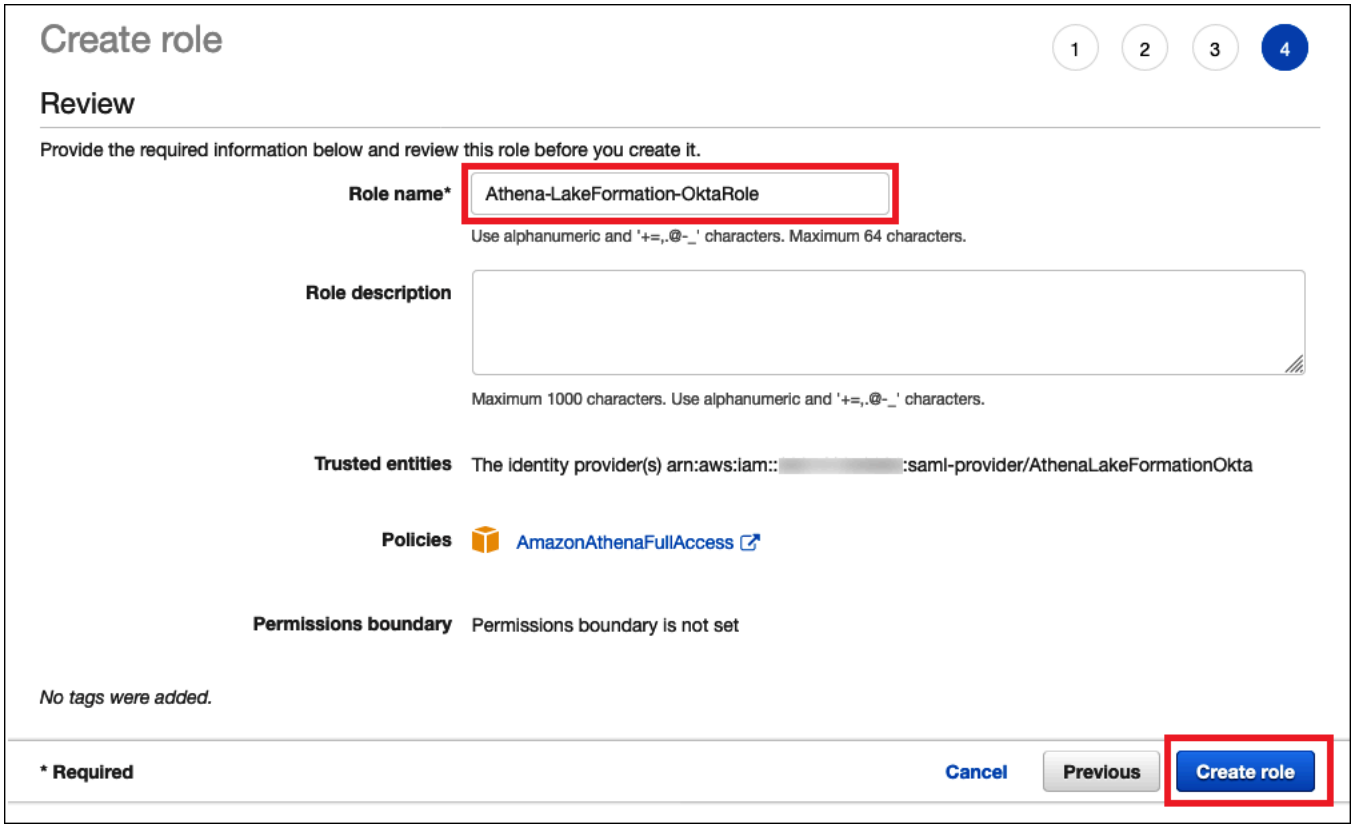
   a. Pour Select type of trusted entity (Sélectionner le type d'entité de confiance), choisissez SAML 2.0 Federation.
   b. Pour le fournisseur SAML, sélectionnez AthenaLakeFormationOkta.
   c. Pour le fournisseur SAML, sélectionnez l'option Autoriser la programmation et AWS Management Console l'accès.

d. Sélectionnez Next: Permissions (Étape suivante : autorisations).

3. Sur la page Attach Permissions policies (Attacher les politiques d'autorisations), pour Filter policies (Politiques de filtrage), saisissez **`Athena`**.
4. Sélectionnez la politique AmazonAthenaFullAccessgérée, puis choisissez Next : Tags.



5. Sur la page Add tags (Ajouter des identifications), choisissez Next: Review (Suivant : Révision).
6. Sur la page Révision, dans Nom du rôle, entrez un nom pour le rôle (par exemple, *`Athéna-LakeFormation - OktaRole`*), puis choisissez Créer un rôle.



Ensuite, vous ajoutez des politiques intégrées qui autorisent l'accès à Lake Formation, aux AWS Glue API et aux résultats des requêtes Athena dans Amazon S3.

Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Pour ajouter une politique intégrée au rôle de Lake Formation et AWS Glue

1. Dans la liste des rôles de la console IAM, choisissez le rôle `Athena-LakeFormation-OktaRole` nouvellement créé.
2. Sur la page Summary (Résumé) du rôle, dans l'onglet Permissions (Autorisations), choisissez Add inline policy (Ajouter une politique en ligne).
3. Sur la page Créer une politique, choisissez JSON.
4. Ajoutez une politique en ligne comme la suivante qui donne accès à Lake Formation et aux API AWS Glue .

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

```
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess",
            "glue:GetTable",
            "glue:GetTables",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:CreateDatabase",
            "glue:GetUserDefinedFunction",
            "glue:GetUserDefinedFunctions"
        ],
        "Resource": "*"
    }
}
```

5. Choisissez Review policy (Examiner une politique).
6. Dans le champ Name (Nom), saisissez un nom pour la politique (par exemple, **`LakeFormationGlueInlinePolicy`**).
7. Choisissez Créer une politique.

Ajout d'une politique en ligne au rôle pour l'emplacement des résultats des requêtes Athena

1. Sur la page Summary (Résumé) du rôle `Athena-LakeFormation-OktaRole`, dans l'onglet Permissions (Autorisations), choisissez Add inline policy (Ajouter une politique en ligne).
2. Sur la page Créer une politique, choisissez JSON.
3. Ajoutez une politique en ligne comme la suivante qui autorise le rôle à accéder à l'emplacement des résultats des requêtes Athena. Remplacez les espaces réservés *`< athena-query-results-bucket >`* dans l'exemple par le nom de votre compartiment Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AthenaQueryResultsPermissionsForS3",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:PutObject",
                "s3:GetObject"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::<athena-query-results-bucket>",
                "arn:aws:s3:::<athena-query-results-bucket>/*"
            ]
        }
    ]
}
```

4. Choisissez Review policy (Examiner une politique).
5. Dans le champ Name (Nom), saisissez un nom pour la politique (par exemple, **`AthenaQueryResultsInlinePolicy`**).
6. Choisissez Créer une politique.

Ensuite, vous copiez l'ARN du rôle d'accès de Lake Formation et l'ARN du fournisseur SAML que vous avez créé. Ils sont nécessaires lorsque vous configurez l'application Okta SAML dans la section suivante du tutoriel.

Copier le rôle ARN et ARN du fournisseur d'identité SAML

1. Dans la console IAM, sur la page Summary (Résumé) pour le rôle `Athena-LakeFormation-OktaRole`, cliquez sur l'icône Copy to clipboard (Copier dans le presse-papiers) à côté de Role ARN (ARN de rôle). L'ARN a le format suivant :

   ```
   arn:aws:iam::<account-id>:role/Athena-LakeFormation-OktaRole
   ```

2. Enregistrez l'ARN complet en toute sécurité pour référence ultérieure.
3. Dans le panneau de navigation de la console IAM, choisissez Identity providers (Fournisseurs d'identité).
4. Choisissez le AthenaLakeFormationOktafournisseur.
5. Sur la page Summary (Résumé), cliquez sur l'icône Copy to clipboard (Copier dans le presse-papiers) à côté de Provider ARN (ARN du fournisseur). L'ARN doit ressembler à l'exemple suivant :

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta
   ```

6. Enregistrez l'ARN complet en toute sécurité pour référence ultérieure.

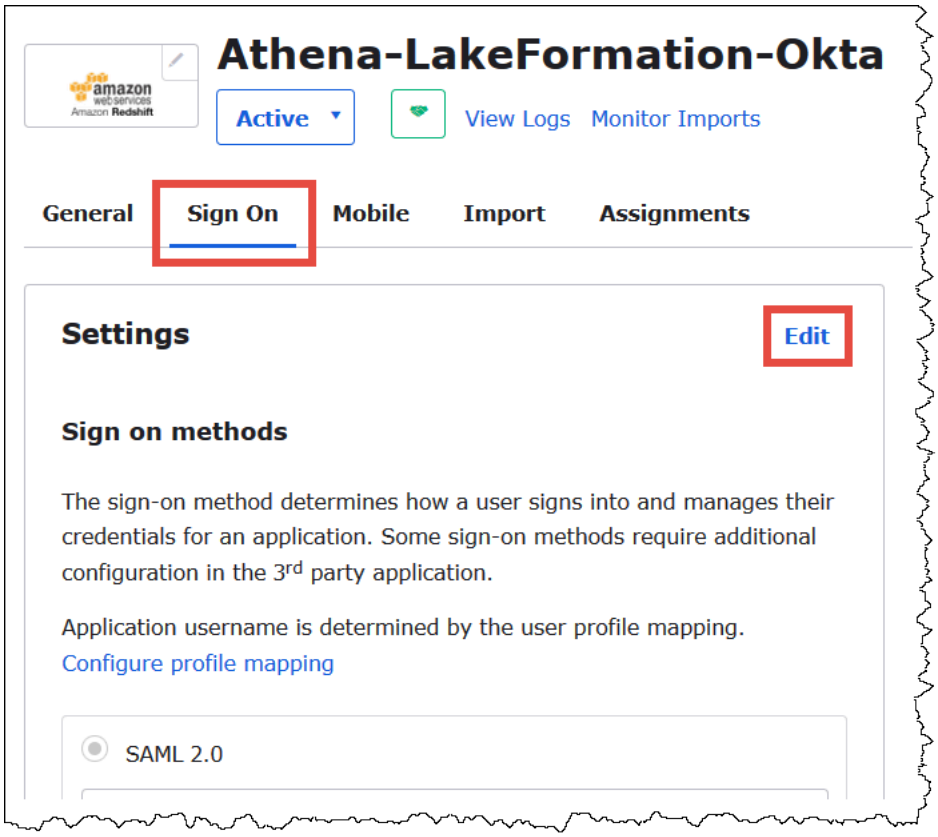
Étape 5 : ajouter le rôle IAM et le fournisseur d'identité SAML à l'application Okta

Dans cette étape, vous revenez à la console du développeur Okta et effectuez les tâches suivantes :

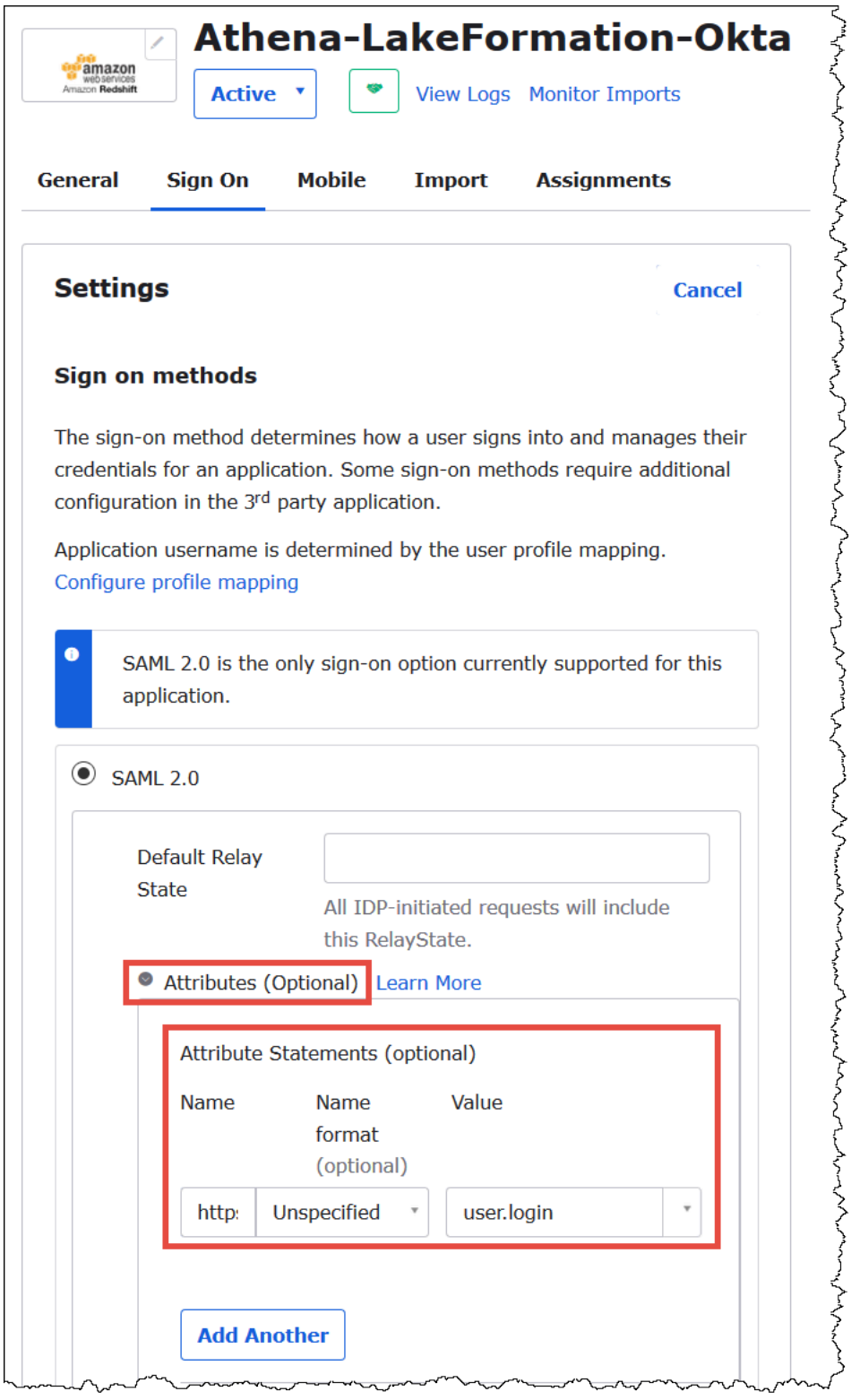
- Ajouter les attributs URL Lake Formation de l'utilisateur et du groupe à l'application Okta.
- Ajouter l'ARN pour le fournisseur d'identité et l'ARN pour le rôle IAM à l'application Okta.
- Copier l'ID de l'application Okta. L'ID de l'application Okta est requis dans le profil JDBC qui se connecte à Athena.

Ajout d'attributs d'URL Lake Formation de l'utilisateur et du groupe à l'application Okta

1. Connectez-vous à la console de développement Okta.
2. Choisissez l'onglet Applications, puis choisissez l'application `Athena-LakeFormation-Okta`.
3. Choisissez l'onglet Sign On (Authentification) de l'application, puis choisissez Edit (Modifier).



4. Choisissez Attributes (optional) (Attributs [facultatif]) pour les développer.



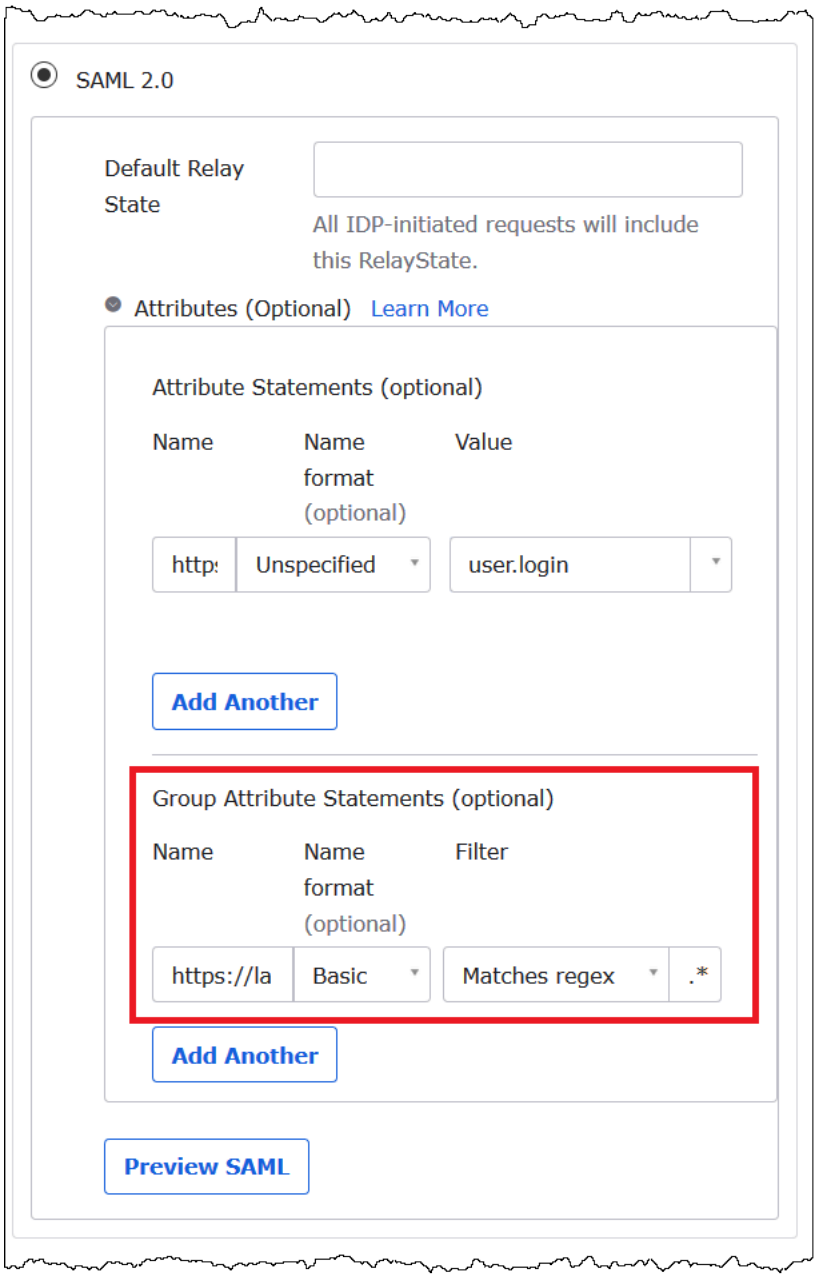
5. Pour Attribute Statements (optional) (Instructions d'attribut [facultatif]), ajoutez l'attribut suivant :

   - Pour Name (Nom), saisissez **`https://lakeformation.amazon.com/SAML/Attributes/Username`**.

- Pour le champ Value (Valeur), saisissez **`user.login`**.

6. Sous Group Attribute Statements (optional) (Instructions d'attribut de groupe [facultatif]), ajoutez l'attribut suivant :

   - Pour Name (Nom), saisissez **`https://lakeformation.amazon.com/SAML/Attributes/Groups`**.
   - Pour Name format (Format du nom), saisissez **`Basic`**
   - Pour Filter (Filtre), choisissez Matches regex (Correspond à regex), puis saisissez **`.*`** dans la zone de filtre.

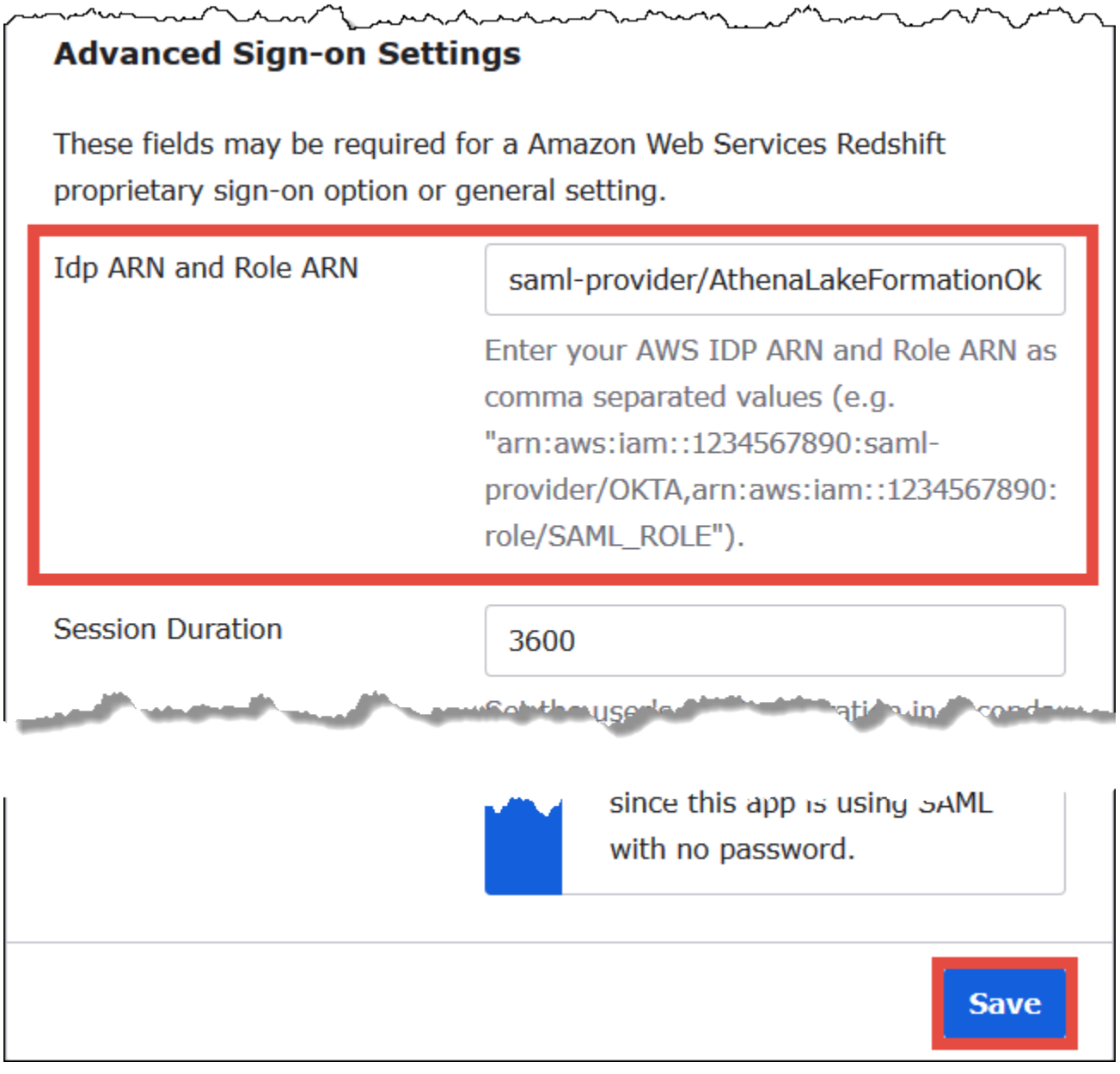


7. Faites défiler vers le bas jusqu'à la section Advanced Sign-On Settings (Paramètres avancés d'ouverture de session), où vous ajouterez les ARN du fournisseur d'identité et du rôle IAM à l'application Okta.

Ajout des ARN pour le fournisseur d'identité et pour le rôle IAM à l'application Okta

1. <saml-arn><role-arn>Pour l'ARN Idp et l'ARN du rôle, entrez l'ARN du fournisseur AWS d'identité et l'ARN du rôle sous forme de valeurs séparées par des virgules au format,. La chaîne combinée devrait ressembler à ce qui suit :

```
arn:aws:iam::<account-id>:saml-provider/
AthenaLakeFormationOkta,arn:aws:iam::<account-id>:role/Athena-LakeFormation-
OktaRole
```

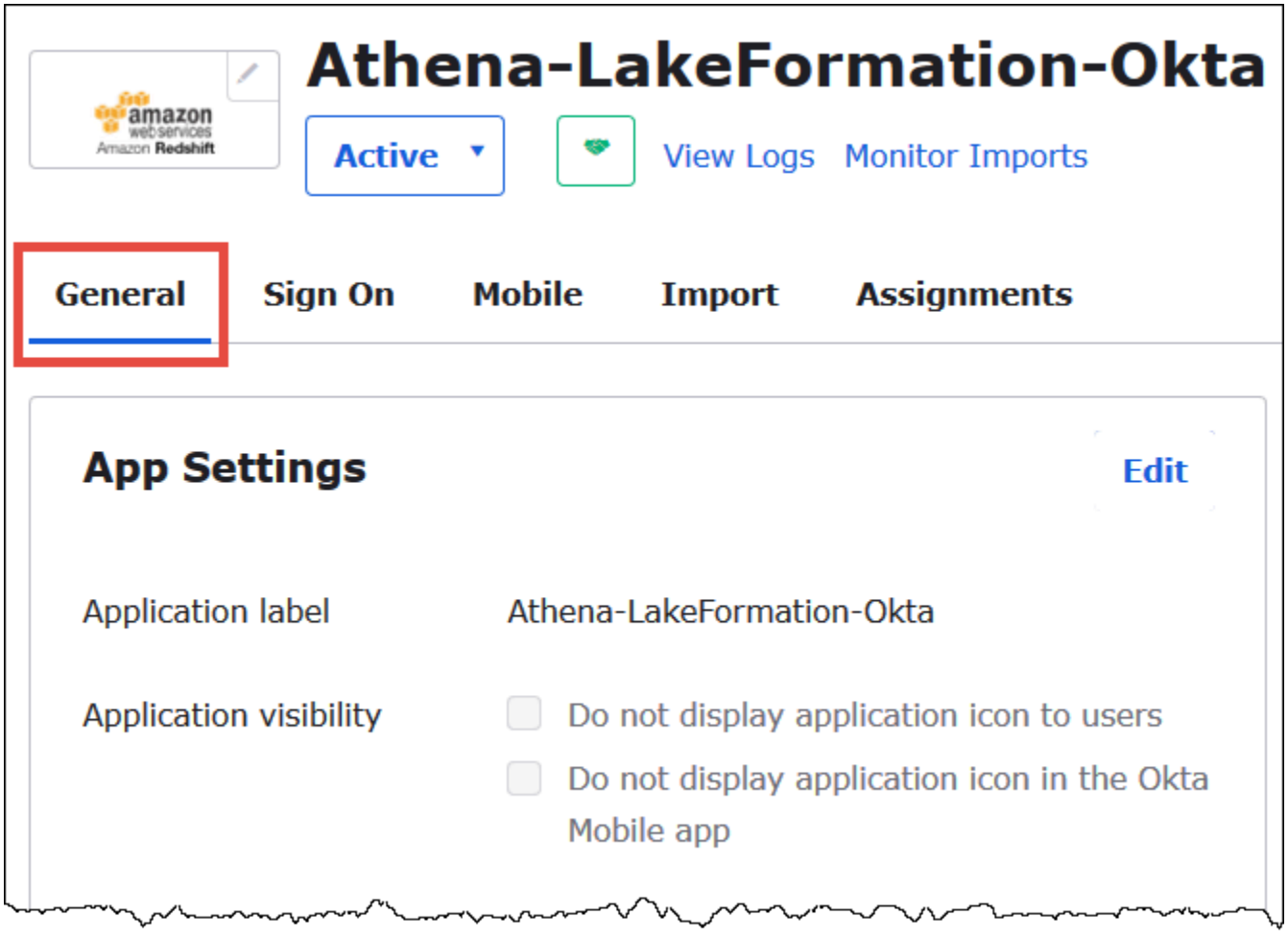


2. Choisissez Enregistrer.

Ensuite, copiez l'ID de l'application Okta. Vous en aurez besoin plus tard pour la chaîne JDBC qui se connecte à Athena.

Recherche et copie de l'ID de l'application Okta

1. Choisissez l'onglet General (Général) de l'application Okta.



2. Faites défiler la page jusqu'à la section App Embed Link (Intégrer lien d'application).
3. À partir de Embed Link (Intégrer lien), copiez et enregistrez de manière sécurisée la partie de l'URL contenant l'ID de l'application Okta. L'ID de l'application Okta est la partie de l'URL après `amazon_aws_redshift/`, mais avant la prochaine barre oblique. Par exemple, si l'URL contient `amazon_aws_redshift/aaa/bbb`, l'ID de l'application est `aaa`.

> **Note**
>
> Le lien intégré ne peut pas être utilisé dans le but de se connecter directement à la console Athena pour afficher les bases de données. Les autorisations Lake Formation pour les utilisateurs et les groupes SAML sont reconnues uniquement lorsque vous utilisez le pilote JDBC ou ODBC pour envoyer des requêtes à Athena. Pour afficher les bases de données, vous pouvez utiliser l'outil SQL Workbench/J, qui utilise le pilote JDBC pour se connecter à Athena. L'outil SQL Workbench/J est couvert par Étape 7 : Vérifier l'accès via le client JDBC d'Athena.

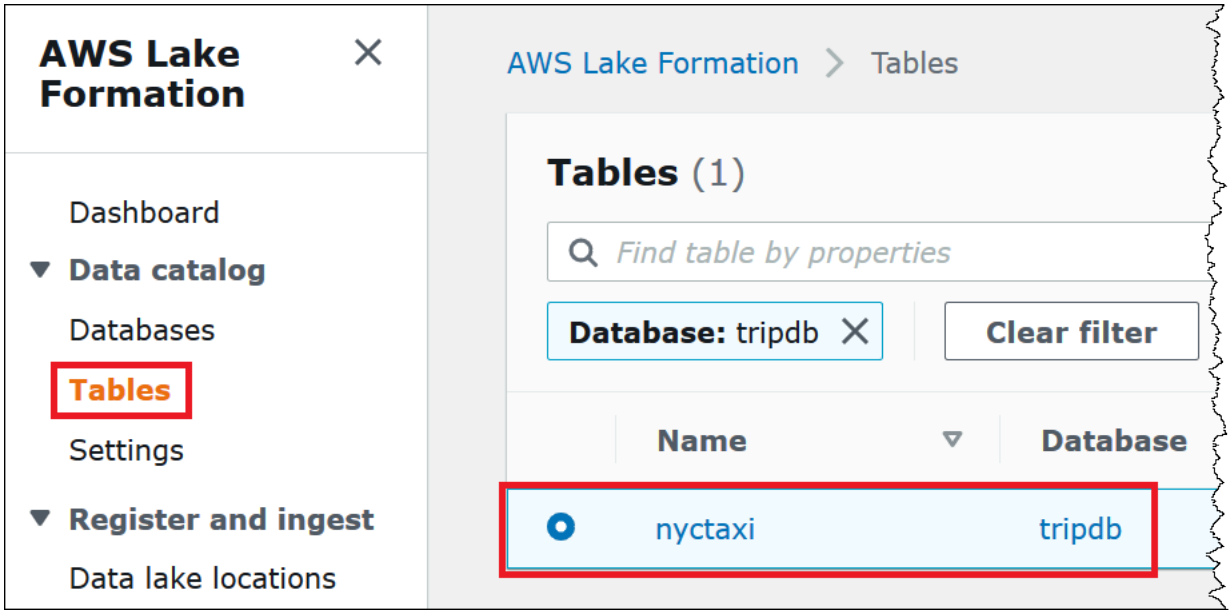
Étape 6 : Accorder des autorisations aux utilisateurs et aux groupes via AWS Lake Formation

Dans cette étape, vous utilisez la console Lake Formation pour accorder des autorisations sur une table à l'utilisateur et au groupe SAML. Vous devez effectuer les tâches suivantes :

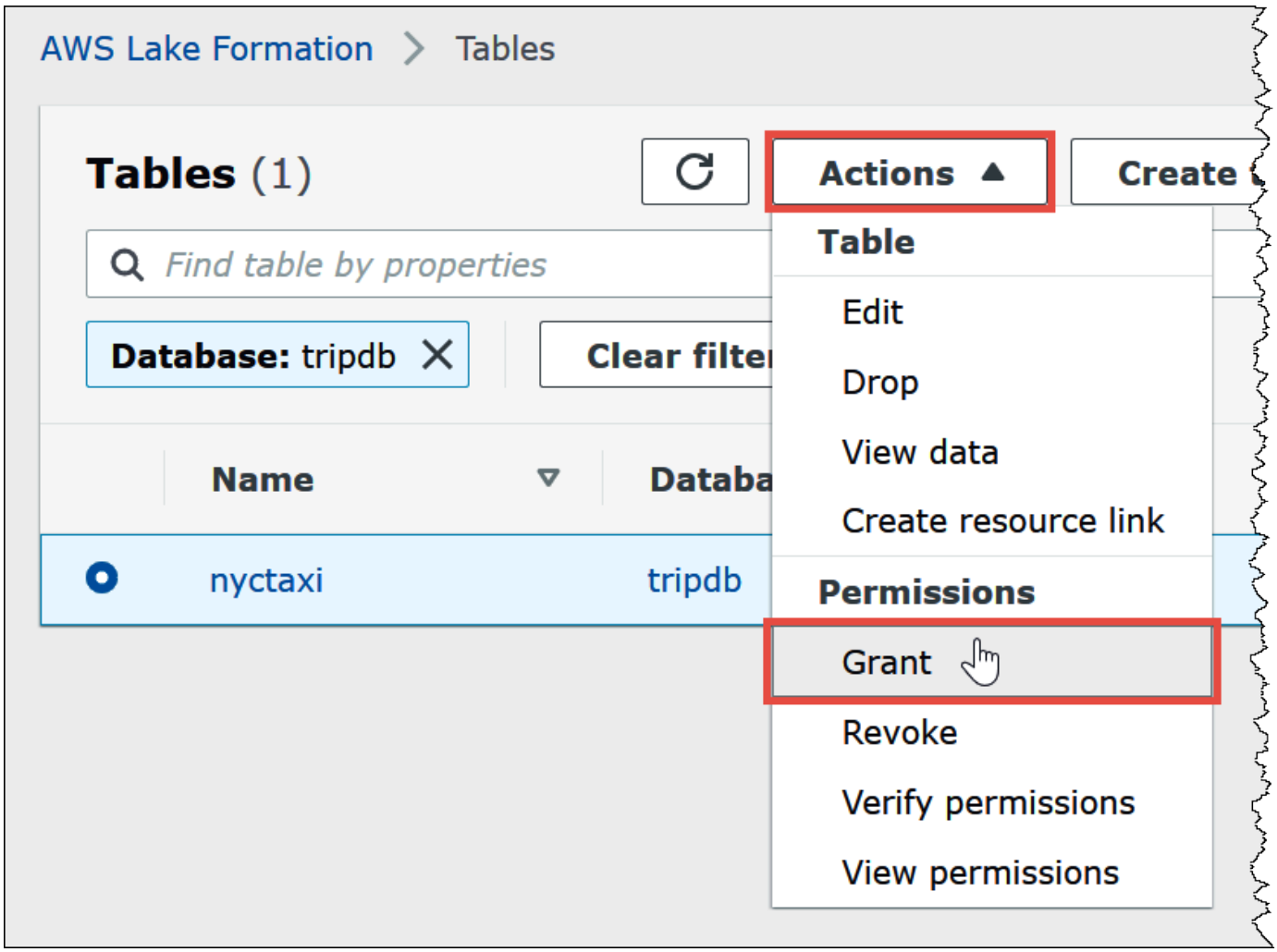
- Spécifier l'ARN de l'utilisateur Okta SAML et les autorisations d'utilisateur associées sur la table.
- Spécifier l'ARN du groupe Okta SAML et les autorisations de groupe associées sur la table.
- Vérifiez les autorisations que vous avez accordées.

Octroi d'autorisations dans Lake Formation à l'utilisateur Okta

1. Connectez-vous à la AWS Management Console en tant qu'administrateur du lac de données.
2. Ouvrez la console Lake Formation à l'adresse https://console.aws.amazon.com/lakeformation/.
3. Dans le panneau de navigation, choisissez Tables, puis sélectionnez la table pour laquelle vous souhaitez accorder des autorisations. Ce tutoriel utilise la table `nyctaxi` de la base de données `tripdb`.



4. Dans Actions, choisissez Grant (Accorder).



5. Dans la boîte de dialogue Grant permissions (Accorder des autorisations), saisissez les informations suivantes :

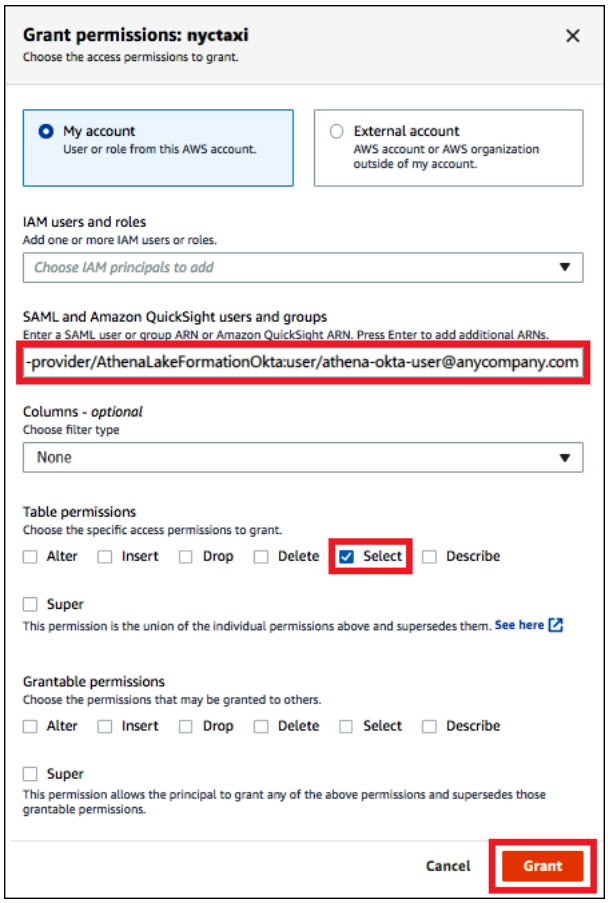
   a. Sous QuickSight Utilisateurs et groupes SAML et Amazon, entrez l'ARN de l'utilisateur Okta SAML au format suivant :

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta:user/<athena-okta-user>@<anycompany.com>
   ```

   b. Pour Columns (Colonnes), pour Choose filter type (Choisir un type de filtre), choisissez Include columns (Inclure des colonnes) ou Exclude columns (Exclure des colonnes).

   c. Utilisez la liste déroulante Choose one or more columns (Choisir une ou plusieurs colonnes) sous le filtre pour spécifier les colonnes que vous voulez inclure ou exclure pour ou de l'utilisateur.

d. Pour Table permissions (Autorisations de table), choisissez Select (Sélectionner). Ce tutoriel n'accorde que l'autorisation SELECT ; vos besoins peuvent varier.



6. Choisissez Grant (Accorder).

Vous devez maintenant effectuer des étapes similaires pour le groupe Okta.

Octroi d'autorisations dans Lake Formation au groupe Okta

1. Sur la page Tables de la console Lake Formation, assurez-vous que la table nyctaxi est toujours sélectionnée.
2. Dans Actions, choisissez Grant (Accorder).
3. Dans la boîte de dialogue Grant permissions (Accorder des autorisations), saisissez les informations suivantes :

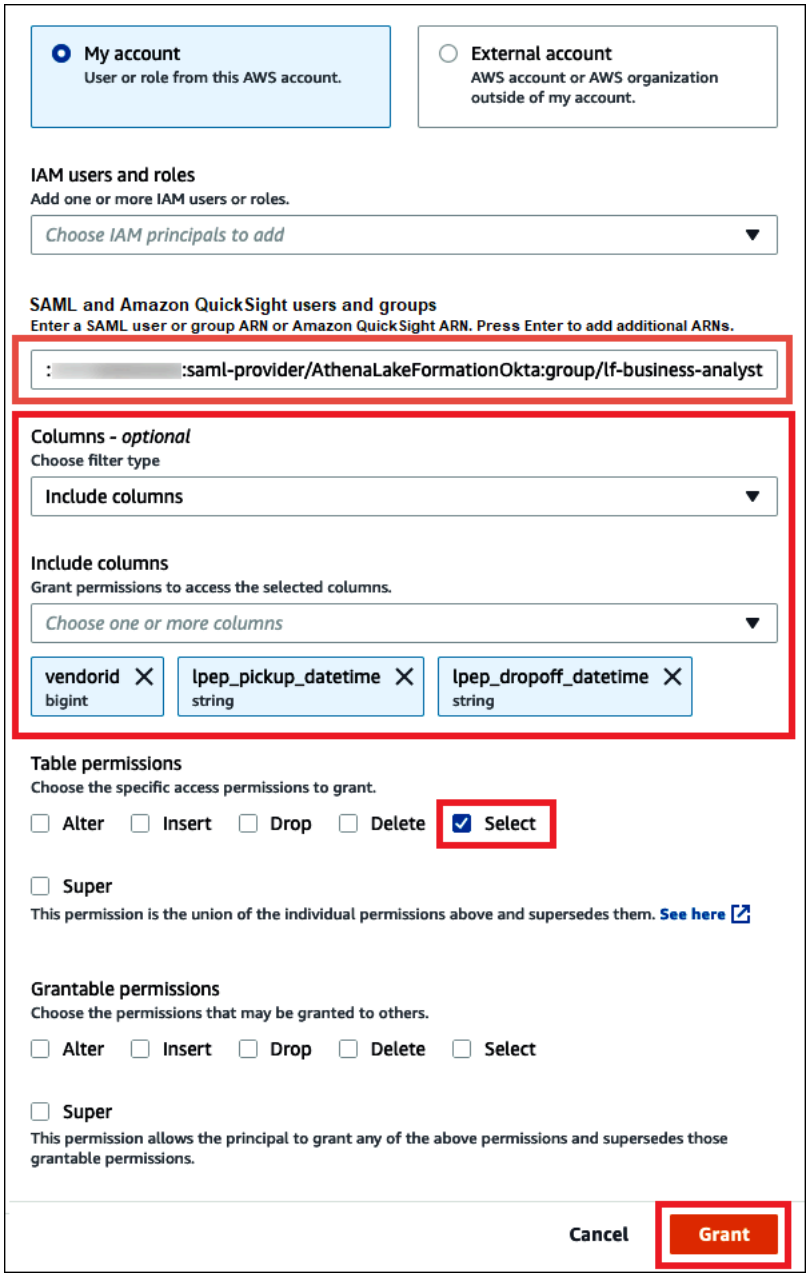
   a. Sous QuickSight Utilisateurs et groupes SAML et Amazon, entrez l'ARN du groupe Okta SAML au format suivant :

```
arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta:group/lf-
business-analyst
```

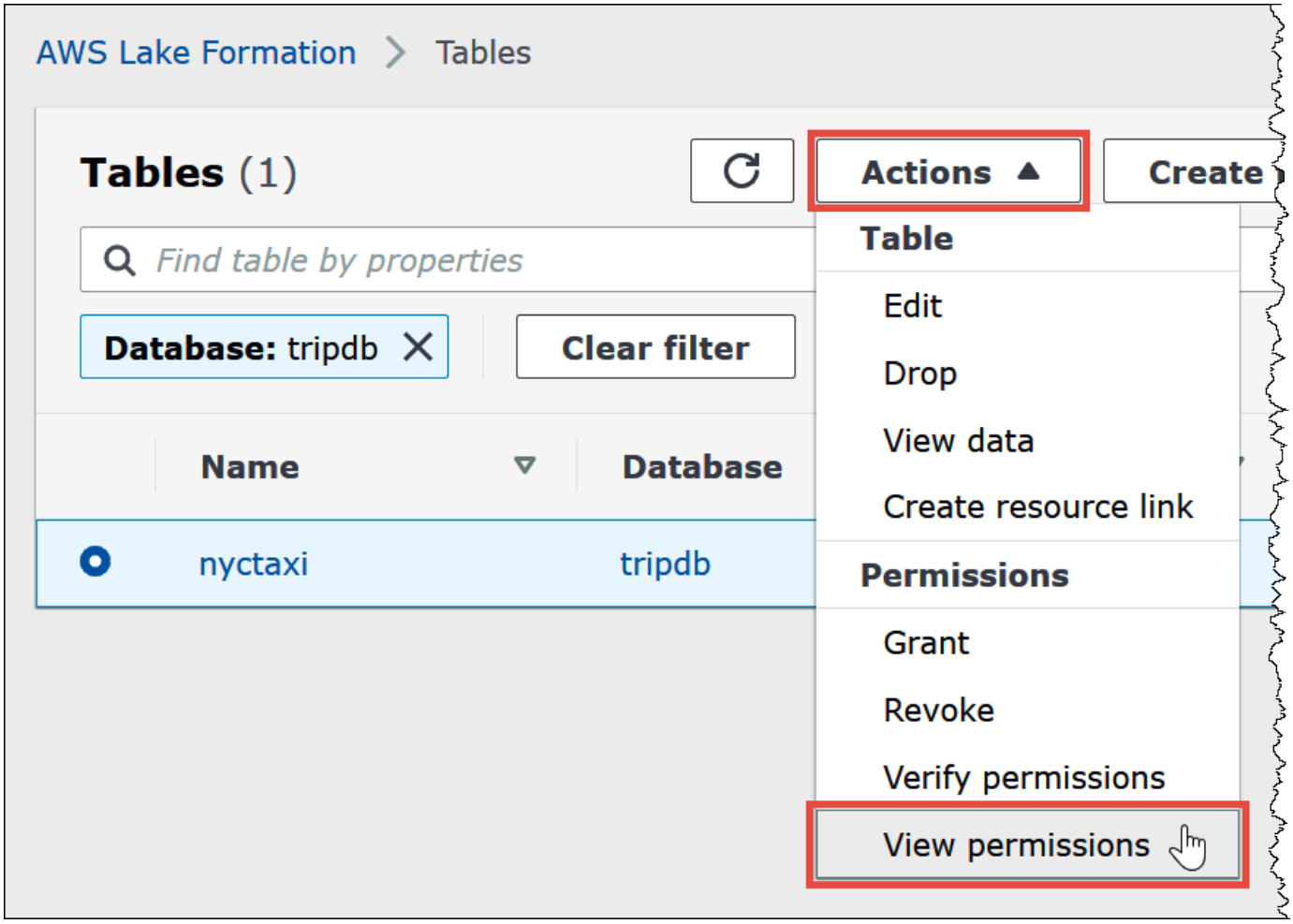
b. Pour Columns (Colonnes), Choose filter type (Choisir un type de filtre), choisissez Include columns (Inclure des colonnes).

c. Pour Choose one or more columns (Choisir une ou plusieurs colonnes), choisissez les trois premières colonnes de la table.

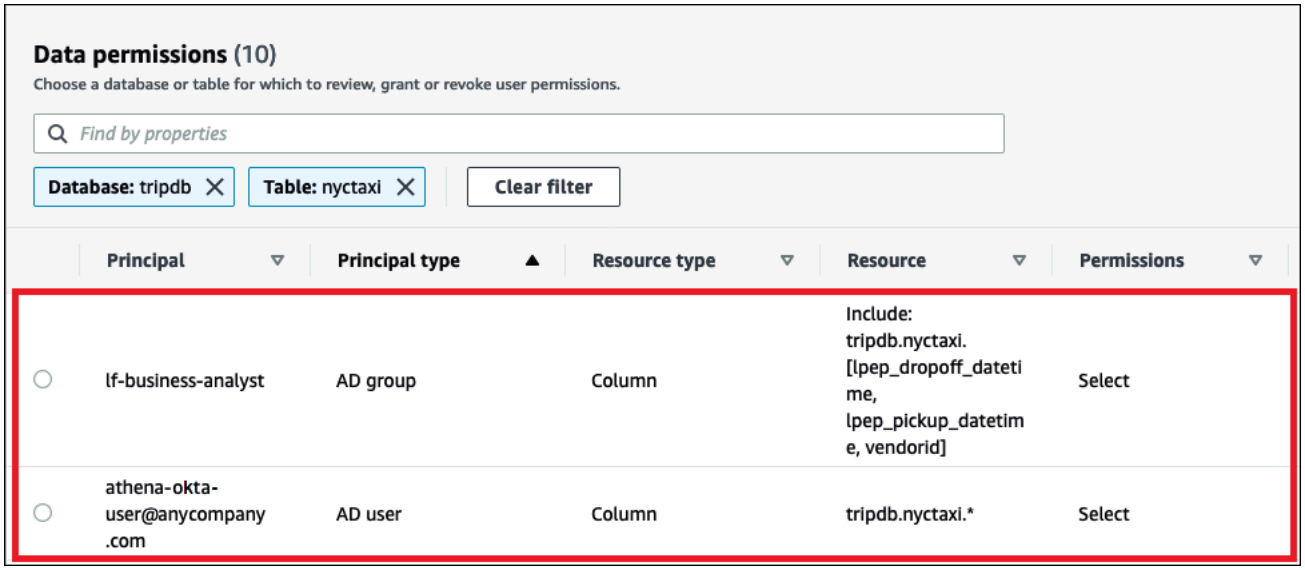
d. Pour Table permissions (Autorisations de table), choisissez les autorisations d'accès spécifiques à accorder. Ce tutoriel n'accorde que l'autorisation `SELECT` ; vos besoins peuvent varier.



4. Choisissez Grant (Accorder).
5. Pour vérifier les autorisations que vous avez accordées, choisissez Actions, View permissions (Afficher les autorisations).



La page Autorisations relatives aux données du `nyctaxi` tableau indique les autorisations pour athena-okta-useret le lf-business-analystgroupe.



Étape 7 : Vérifier l'accès via le client JDBC d'Athena

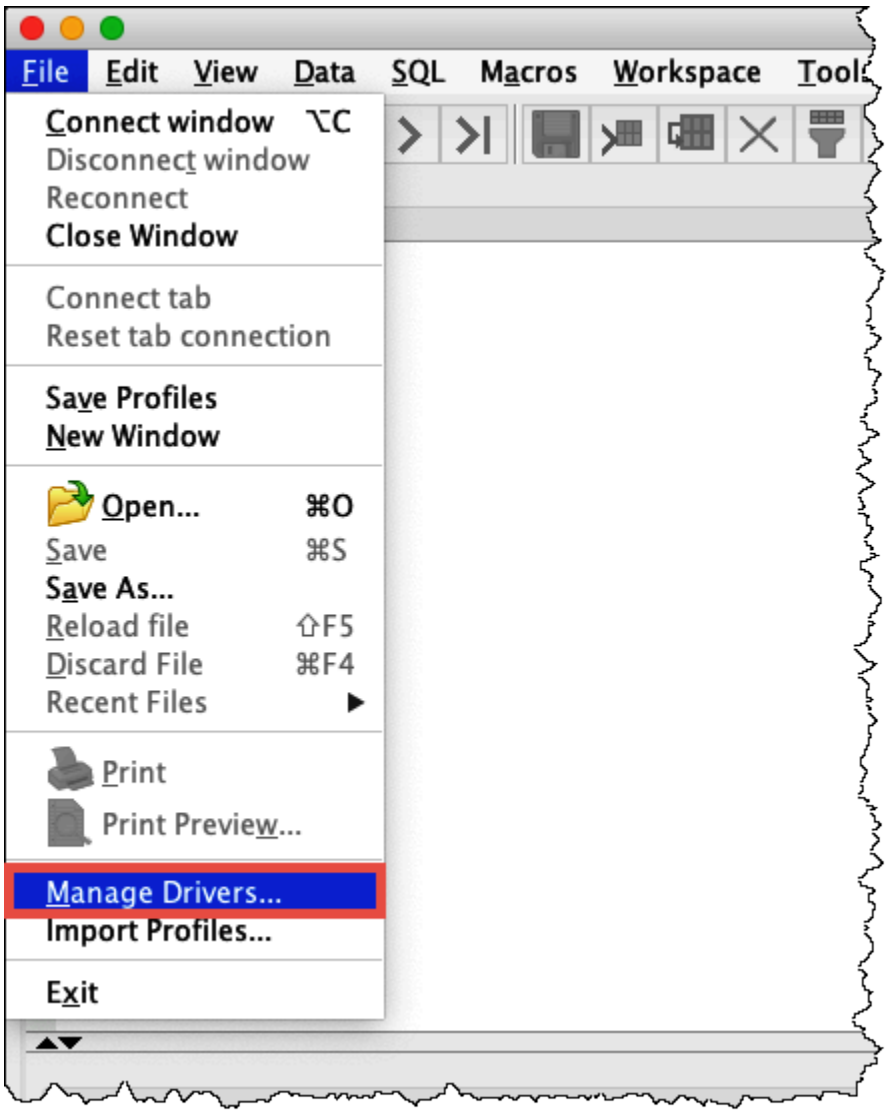
Vous pouvez maintenant utiliser un client JDBC pour effectuer une connexion de test à Athena en tant qu'utilisateur Okta SAML.

Dans cette section, vous effectuez les tâches suivantes :

- Préparer le client de test : téléchargez le pilote JDBC d'Athena, installez SQL Workbench et ajoutez le pilote à Workbench. Ce tutoriel utilise SQL Workbench pour accéder à Athena via l'authentification Okta et pour vérifier les autorisations Lake Formation.
- Dans SQL Workbench :
  - Créez une connexion pour l'utilisateur Okta Athena.
  - Exécutez des requêtes de test en tant qu'utilisateur Okta Athena.
  - Créez et testez une connexion pour l'utilisateur analyste commercial.
- Dans la console Okta, ajoutez l'utilisateur analyste commercial au groupe développeur.
- Dans la console Lake Formation, configurez les autorisations de table pour le groupe développeur.
- Dans SQL Workbench, exécutez des requêtes de test en tant qu'utilisateur analyste commercial et vérifiez comment la modification des autorisations affecte les résultats.

Préparation du client de test

1. Téléchargez et extrayez le pilote JDBC Athena compatible Lake Formation (version 2.0.14 ou ultérieure) à partir de Connexion à Amazon Athena avec JDBC.
2. Téléchargez et installez l'outil d'interrogation SQL gratuit SQL Workbench/J, disponible sous une licence Apache 2.0 modifiée.
3. Dans SQL Workbench/J, choisissez File (Fichier), puis Manage Drivers (Gérer les pilotes).



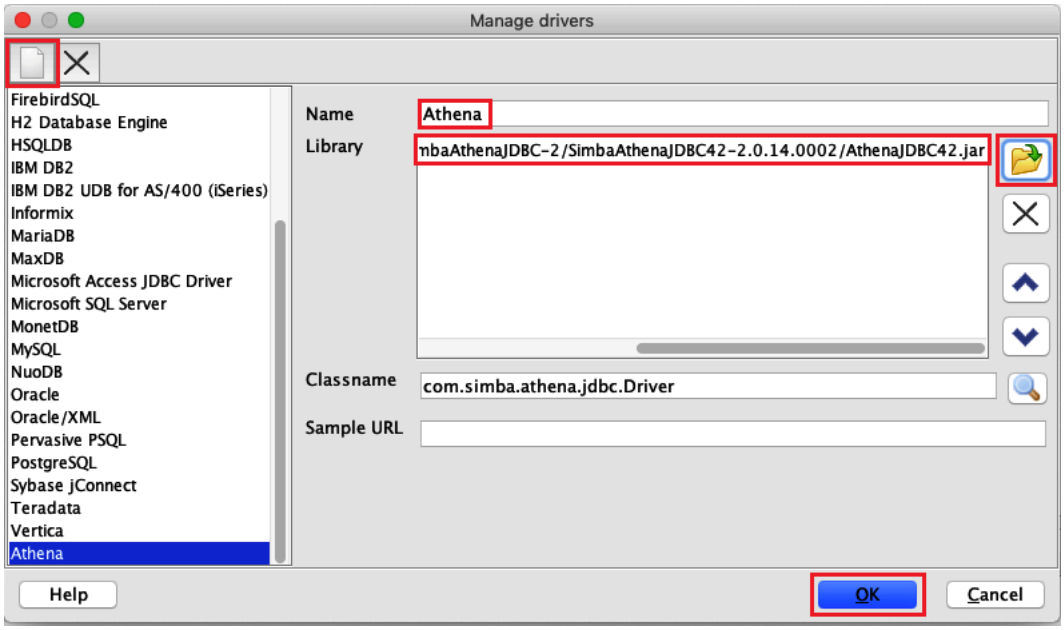
4. Dans la boîte de dialogue Manage Drivers (Gérer les pilotes), effectuez les étapes suivantes :

   a. Choisissez l'icône du nouveau pilote.

   b. Pour Name (Nom), saisissez **`Athena`**.

   c. Pour Library (Bibliothèque), recherchez et choisissez le fichier Simba Athena JDBC `.jar` que vous venez de télécharger.

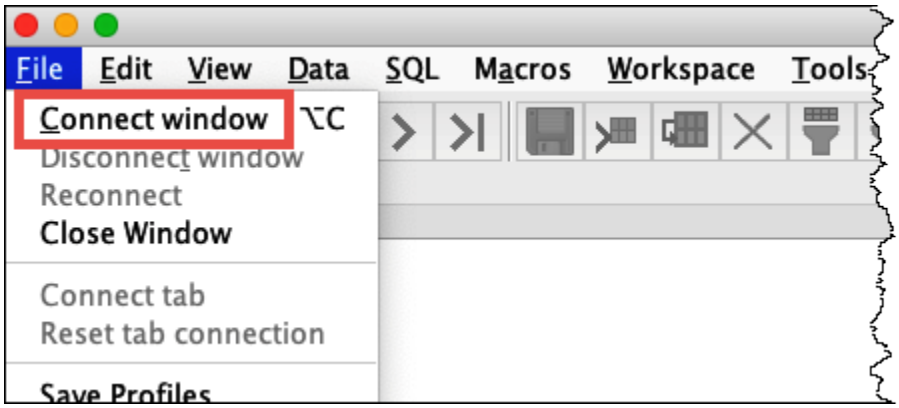
   d. Choisissez OK.



Vous pouvez maintenant créer et tester une connexion pour l'utilisateur Okta Athena.

Création d'une connexion pour l'utilisateur Okta Athena

1. Choisissez File (Fichier), Connect window (Fenêtre de connexion).



2. Dans la boîte de dialogue Connection profile (Profil de connexion), créez une connexion en saisissant les informations suivantes :

   - Dans le champ du nom, saisissez **`Athena_Okta_User_Connection`**.
   - Pour Driver (Pilote), choisissez le pilote JDBC Simba Athena.
   - Pour URL, effectuez l'une des actions suivantes :
     - Pour utiliser une URL de connexion, saisissez une chaîne de connexion d'une seule ligne. L'exemple suivant ajoute des sauts de ligne pour plus de lisibilité.

```
jdbc:awsathena://AwsRegion=region-id;
S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
AwsCredentialsProviderClass=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider;
user=athena-okta-user@anycompany.com;
password=password;
idp_host=okta-idp-domain;
App_ID=okta-app-id;
SSL_Insecure=true;
LakeFormationEnabled=true;
```

- Pour utiliser une URL AWS basée sur un profil, effectuez les opérations suivantes :

  1. Configurez un AWS profil doté d'un fichier AWS d'informations d'identification, comme dans l'exemple suivant.

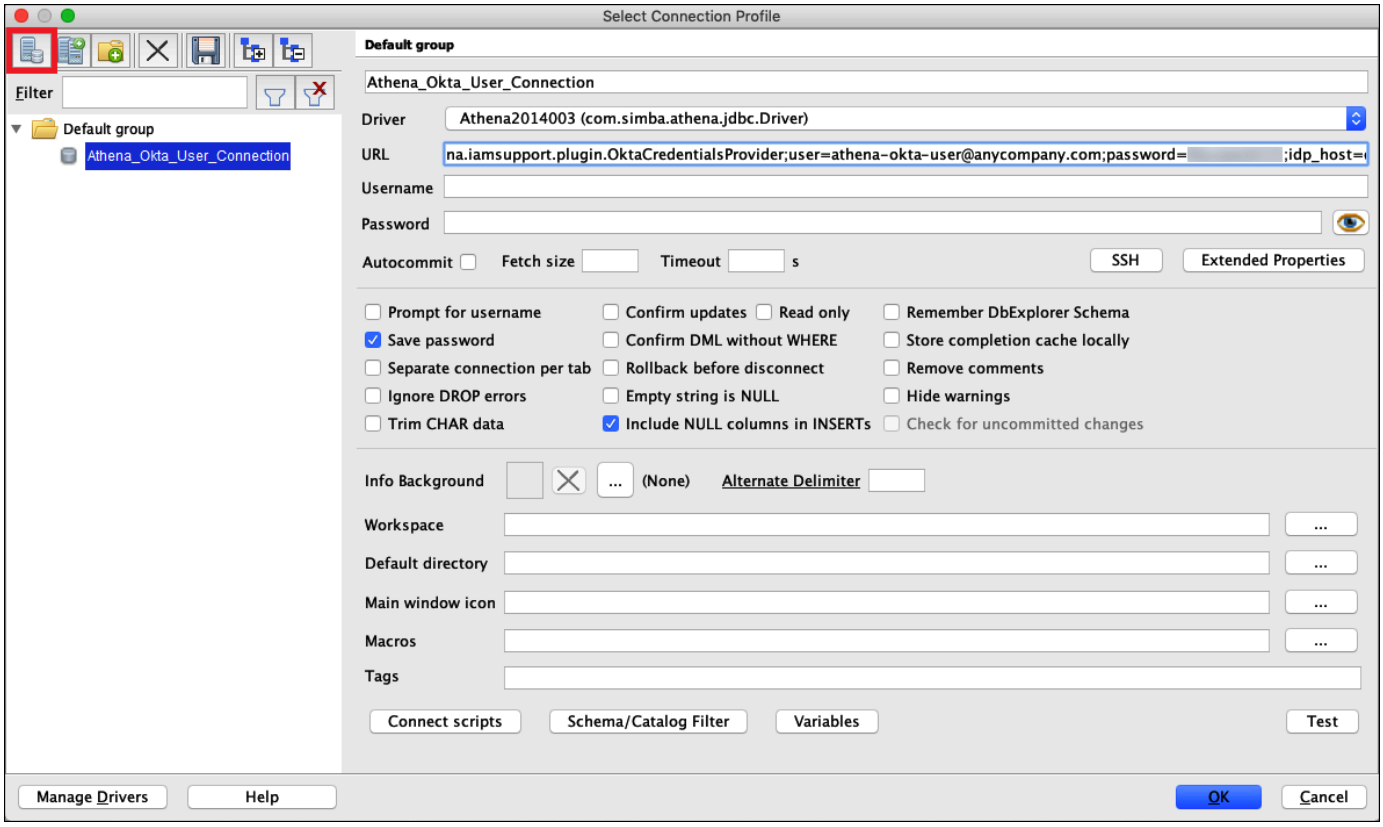
     ```
     [athena_lf_dev]
     plugin_name=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider
     idp_host=okta-idp-domain
     app_id=okta-app-id
     uid=athena-okta-user@anycompany.com
     pwd=password
     ```

  2. Pour URL, saisissez une chaîne de connexion d'une seule ligne comme dans l'exemple suivant. L'exemple ajoute des sauts de ligne pour plus de lisibilité.

     ```
     jdbc:awsathena://AwsRegion=region-id;
     S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
     profile=athena_lf_dev;
     SSL_Insecure=true;
     LakeFormationEnabled=true;
     ```

  Notez que ces exemples sont des représentations de base de l'URL nécessaire pour se connecter à Athena. Pour obtenir la liste complète des paramètres pris en charge dans l'URL, veuillez consulter la documentation JDBC.

L'image suivante montre un profil de connexion SQL Workbench qui utilise une URL de connexion.

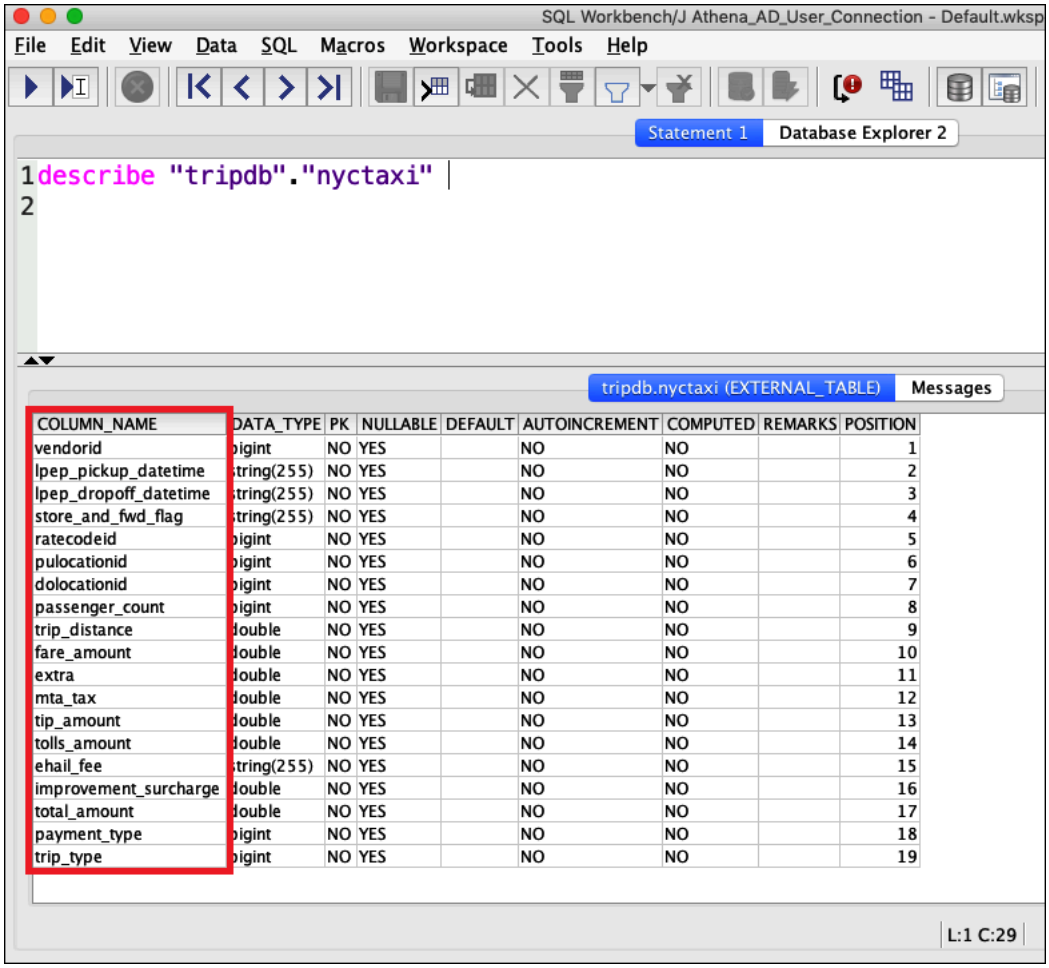


Maintenant que vous avez établi une connexion pour l'utilisateur Okta, vous pouvez la tester en récupérant certaines données.

Test de la connexion de l'utilisateur Okta

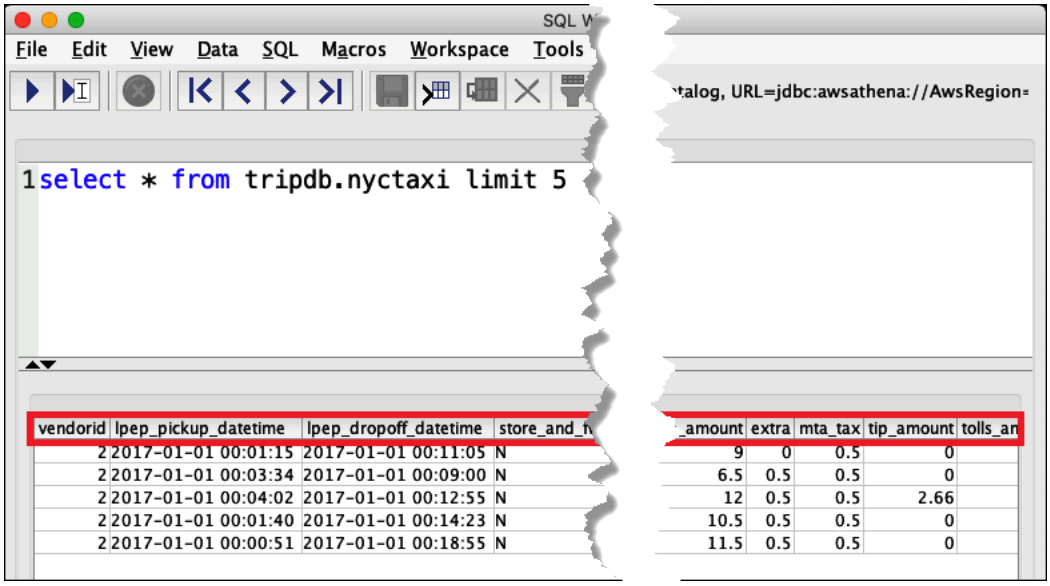
1. Choisissez Test, puis vérifiez que la connexion réussit.
2. Dans la fenêtre Statement (Instruction) de SQL Workbench, exécutez la commande SQL `DESCRIBE` suivante. Vérifiez que toutes les colonnes sont affichées.

```
DESCRIBE "tripdb"."nyctaxi"
```



3. Dans la fenêtre Statement (Instruction) de SQL Workbench, exécutez la commande SQL SELECT suivante. Vérifiez que toutes les colonnes sont affichées.

```
SELECT * FROM tripdb.nyctaxi LIMIT 5
```



Ensuite, vous devez vérifier qu'en tant que membre du lf-business-analystgroupe, il n'a accès qu'aux trois premières colonnes du tableau que vous avez spécifié précédemment dans Lake Formation. athena-ba-user

Pour vérifier l'accès au athena-ba-user

1. Dans SQL Workbench, dans la boîte de dialogue Connection profile (Profil de connexion), créez un autre profil de connexion.

   - Pour le nom du profil de connexion, saisissez **`Athena_Okta_Group_Connection`**.
   - Pour Driver (Pilote), choisissez le pilote JDBC Simba Athena.
   - Pour URL, effectuez l'une des actions suivantes :
     - Pour utiliser une URL de connexion, saisissez une chaîne de connexion d'une seule ligne. L'exemple suivant ajoute des sauts de ligne pour plus de lisibilité.

       ```
       jdbc:awsathena://AwsRegion=region-id;
       S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
       AwsCredentialsProviderClass=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider;
       user=athena-ba-user@anycompany.com;
       password=password;
       idp_host=okta-idp-domain;
       App_ID=okta-application-id;
       SSL_Insecure=true;
       ```

```
LakeFormationEnabled=true;
```

- Pour utiliser une URL AWS basée sur un profil, effectuez les opérations suivantes :

  1. Configurez un AWS profil doté d'un fichier d'informations d'identification, comme dans l'exemple suivant.

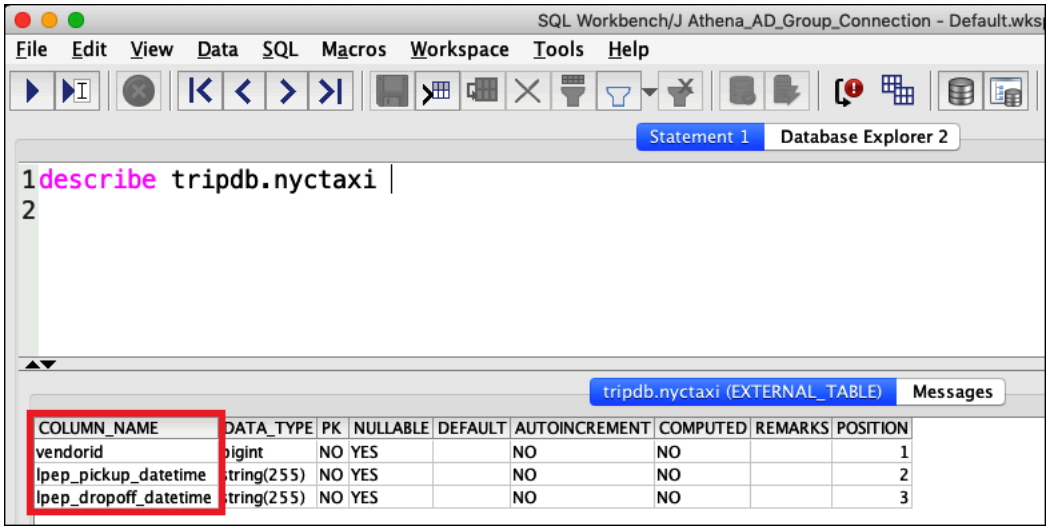
     ```
     [athena_lf_ba]
     plugin_name=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider
     idp_host=okta-idp-domain
     app_id=okta-application-id
     uid=athena-ba-user@anycompany.com
     pwd=password
     ```

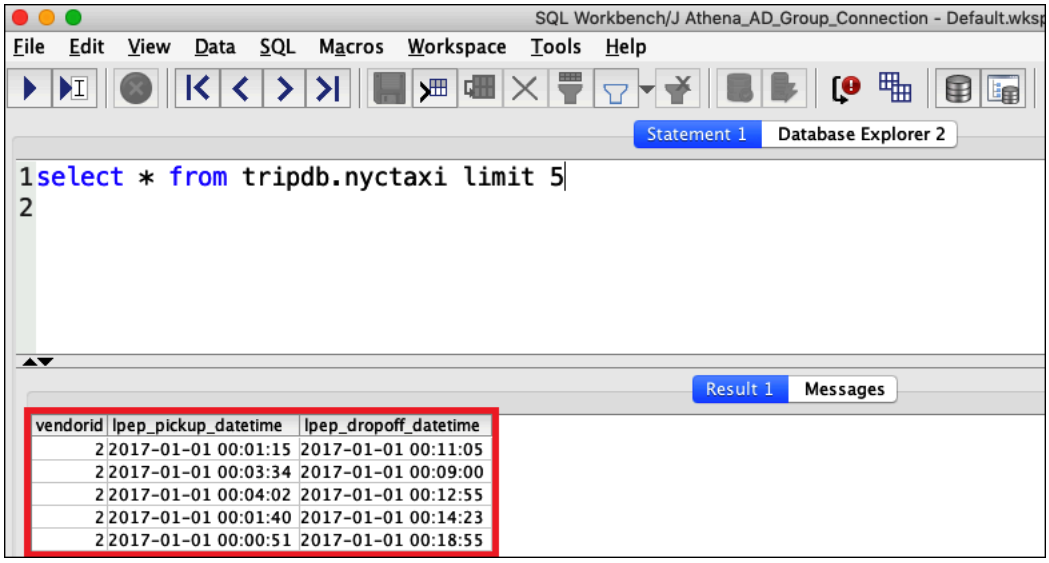
  2. Pour URL, saisissez une chaîne de connexion d'une seule ligne comme la suivante. L'exemple ajoute des sauts de ligne pour plus de lisibilité.

     ```
     jdbc:awsathena://AwsRegion=region-id;
     S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
     profile=athena_lf_ba;
     SSL_Insecure=true;
     LakeFormationEnabled=true;
     ```

2. Choisissez Test pour confirmer que la connexion est réussie.
3. Dans la fenêtre SQL Statement (Instruction SQL), exécutez les mêmes commandes SQL `DESCRIBE` et `SELECT` que précédemment et vérifiez les résultats.

   Comme athena-ba-useril est membre du lf-business-analystgroupe, seules les trois premières colonnes que vous avez spécifiées dans la console Lake Formation sont renvoyées.

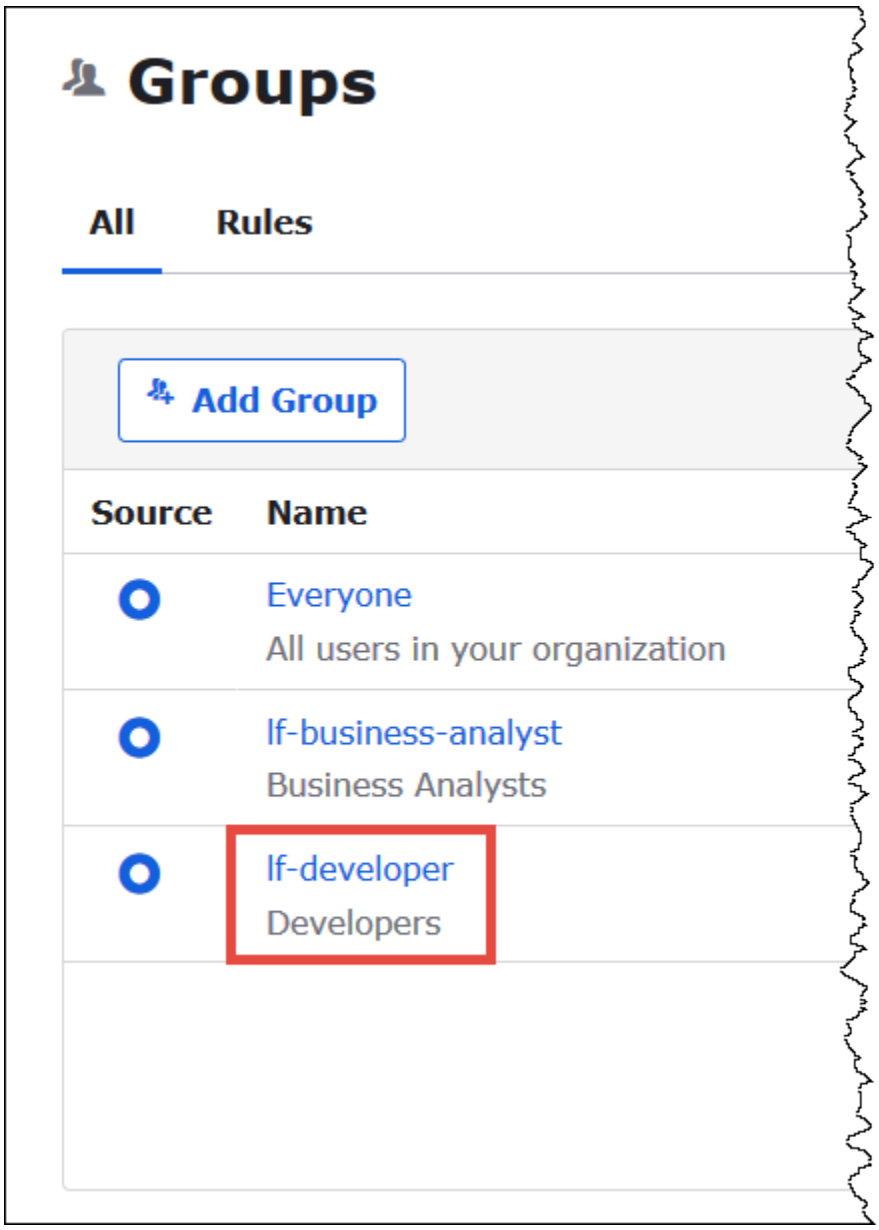




Revenez ensuite à la console Okta pour ajouter l'utilisateur `athena-ba-user` au groupe Okta `lf-developer`.

Pour ajouter le athena-ba-user au groupe lf-developer

1. Connectez-vous à la console Okta en tant qu'utilisateur administratif du domaine Okta attribué.
2. Choisissez Directory (Répertoire), puis choisissez Groups (Groupes).
3. Sur la page Groups (Groupes), choisissez le groupe lf-developer.



4. Choisissez Manage People (Gérer personnes).
5. Dans la liste des non-membres, choisissez le athena-ba-userpour l'ajouter au groupe lf-developer.
6. Choisissez Enregistrer.

Revenez maintenant à la console Lake Formation pour configurer les autorisations de table pour le groupe lf-developer.

Pour configurer les autorisations de table pour lf-developer-group

1. Connectez-vous à la console Lake Formation en tant qu'administrateur de lac de données.
2. Dans le volet de navigation, choisissez Tables.
3. Sélectionnez la table nyctaxi.
4. Choisissez Actions, Grant (Accorder).
5. Dans la boîte de dialogue Grant permissions (Accorder des autorisations), saisissez les informations suivantes :

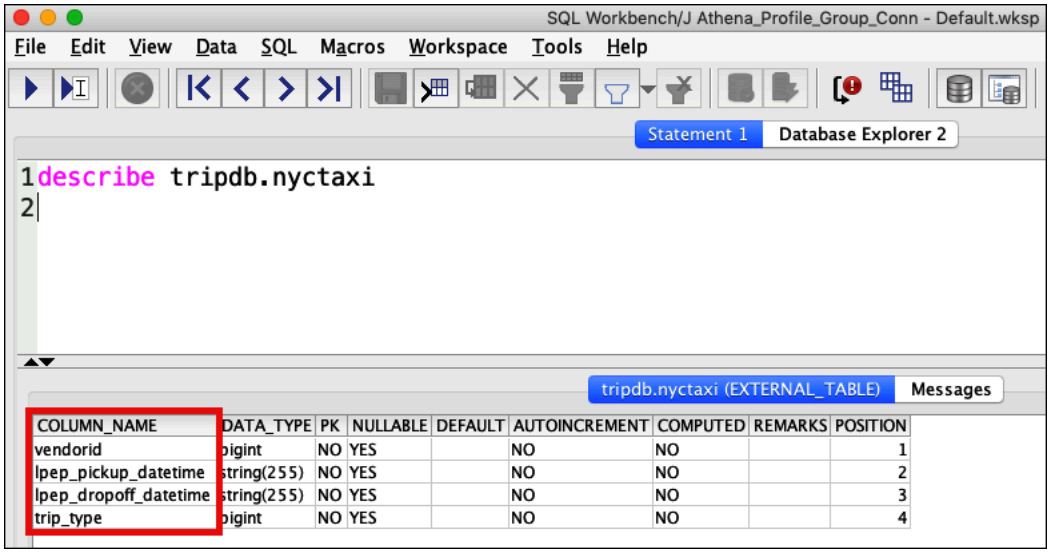
   - Pour les QuickSight utilisateurs et les groupes SAML et Amazon, entrez l'ARN du groupe Okta SAML lf-developer au format suivant :
   - Pour Columns (Colonnes), Choose filter type (Choisir un type de filtre), choisissez Include columns (Inclure des colonnes).
   - Choisissez la colonne trip_type.
   - Pour Table permissions (Autorisations de table), choisissez SELECT (Sélectionner).
6. Choisissez Grant (Accorder).

Vous pouvez maintenant utiliser SQL Workbench pour vérifier la modification des autorisations pour le groupe lf-developer. Le changement devrait se refléter dans les données disponibles pour athena-ba-user, qui est désormais membre du groupe lf-developer.

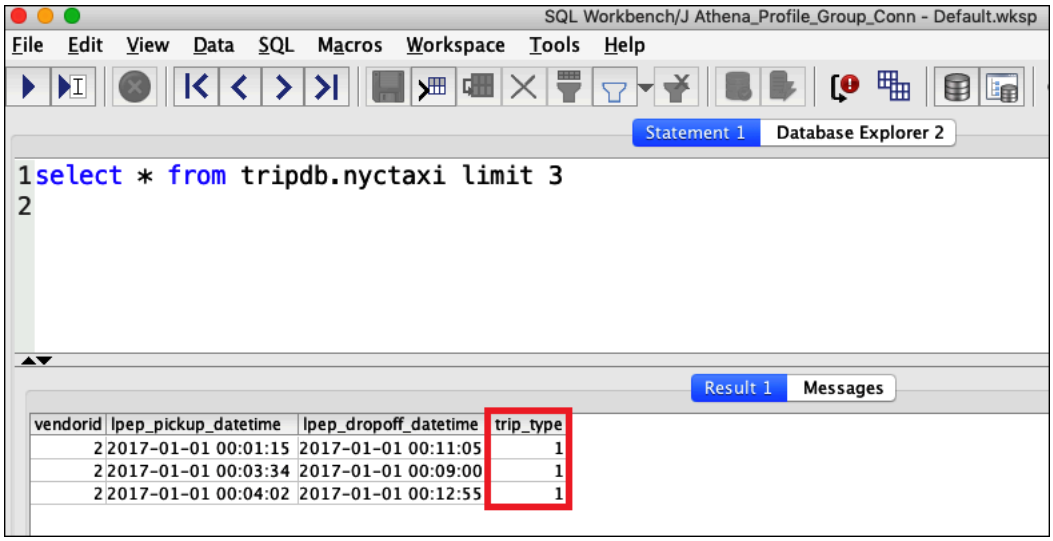
Pour vérifier la modification des autorisations pour athena-ba-user

1. Fermez le programme SQL Workbench, puis rouvrez-le.
2. Connectez-vous au profil pour athena-ba-user.
3. À partir de la fenêtre Statement (Instruction), exécutez les mêmes instructions SQL que précédemment :

   Cette fois, la colonne trip_type s'affiche.



Étant donné qu'athena-ba-useril est désormais membre à la fois du lf-developer et lf-business-analystdes groupes, la combinaison des autorisations Lake Formation pour ces groupes détermine les colonnes renvoyées.



Conclusion

Dans ce didacticiel, vous avez configuré l'intégration d'Athena en AWS Lake Formation utilisant Okta comme fournisseur SAML. Vous avez utilisé Lake Formation et IAM pour contrôler les ressources mises à la disposition de l'utilisateur SAML dans votre catalogue de données de lacs de AWS Glue données.

Ressources connexes

Pour plus d'informations, consultez les ressources suivantes.

- Connexion à Amazon Athena avec JDBC
- Activation de l'accès fédéré à l'API Athena
- AWS Lake Formation Manuel du développeur
- Octroi et révocation des autorisations du catalogue de données dans le Guide du développeur AWS Lake Formation .
- Fournisseurs d'identité et fédération dans le Guide de l'utilisateur IAM.
- Création de fournisseurs d'identité IAM SAML dans le Guide de l'utilisateur IAM.
- Activation de la fédération à AWS l'aide de Windows Active Directory, ADFS et SAML 2.0 sur le blog de AWS sécurité.

Gestion de la charge de travail

Vous pouvez utiliser les fonctionnalités d'Athena relatives aux groupes de travail, à la gestion des capacités, au réglage des performances, à la prise en charge de la compression, aux balises et aux Service Quotas pour gérer votre charge de travail.

Rubriques

- Utilisation des groupes de travail pour contrôler l'accès aux requêtes et les coûts
- Gestion de la capacité de traitement des requêtes
- Réglage de performances dans Athena
- Prise en charge de la compression Athena
- Étiquetage des ressources Athena
- Service Quotas

Utilisation des groupes de travail pour contrôler l'accès aux requêtes et les coûts

Utilisez des groupes de travail pour séparer les utilisateurs, les équipes, les applications ou les charges de travail, pour définir des limites au volume de données pouvant être traité par chaque requête ou groupe de travail entier et pour suivre les coûts. Vous pouvez utiliser des stratégies

basées sur une identité au niveau des ressources pour contrôler l'accès à un groupe de travail spécifique, car les groupes de travail agissent en tant que ressources. Vous pouvez également consulter les métriques relatives aux requêtes dans Amazon CloudWatch, contrôler les coûts en limitant la quantité de données numérisées, créer des seuils et déclencher des actions, telles qu'Amazon SNS, lorsque ces seuils sont dépassés.

Pour mieux contrôler les coûts, vous pouvez créer des réserves de capacité avec le nombre d'unités de traitement de données que vous spécifiez et ajouter un ou plusieurs groupes de travail à la réserve. Pour plus d'informations, consultez Gestion de la capacité de traitement des requêtes.

Les groupes de travail s'intègrent à IAM, à CloudWatch Amazon Simple Notification Service et aux rapports sur les AWS coûts et l'utilisation comme suit :

- Les politiques IAM basées sur l'identité avec des autorisations au niveau des ressources contrôlent qui peut exécuter des requêtes dans un groupe de travail.
- Athena publie les métriques de requête du groupe de travail sur CloudWatch, si vous activez les métriques de requête.
- Dans Amazon SNS, vous pouvez créer des rubriques Amazon SNS qui déclenchent des alarmes pour les utilisateurs spécifiés du groupe de travail lorsque les contrôles d'utilisation des données pour les requêtes dans un groupe de travail dépassent les seuils que vous avez établis.
- Lorsque vous balisez un groupe de travail à l'aide d'une balise configurée en tant que balise de répartition des coûts dans la console de Billing and Cost Management, les coûts associés à l'exécution de requêtes dans ce groupe de travail apparaissent dans vos rapports sur les coûts et l'utilisation avec cette balise de répartition des coûts.

Rubriques

- Utilisation de groupes de travail pour exécuter des requêtes
- Contrôle des coûts et surveillance des requêtes à l'aide de CloudWatch métriques et d'événements

Consultez également le billet de blog AWS Big Data Séparer les requêtes et gérer les coûts à l'aide des groupes de travail Amazon Athena, qui explique comment utiliser les groupes de travail pour séparer les charges de travail, contrôler l'accès des utilisateurs et gérer l'utilisation et les coûts des requêtes.

Utilisation de groupes de travail pour exécuter des requêtes

Nous vous recommandons d'utiliser des groupes de travail pour isoler les requêtes des équipes, des applications ou des charges de travail différentes. Par exemple, vous pouvez créer des groupes de travail distincts pour deux équipes différentes de votre organisation. Vous pouvez également séparer des charges de travail. Par exemple, vous pouvez créer deux groupes de travail indépendants, un pour les applications planifiées automatisées, telles que la génération de rapport, et un autre pour une utilisation ponctuelle par des analystes. Vous pouvez basculer entre les groupes de travail.

Rubriques

Avantages offerts par l'utilisation de groupes de travail

Les groupes de travail vous permettent d'effectuer les opérations suivantes :

Isoler des utilisateurs, des équipes, des applications ou des charges de travail dans des groupes.	Chaque groupe de travail a son propre historique des requêtes et une liste des requêtes enregistrées. Pour plus d'informations, consultez Fonctionnement des groupes de travail. Pour toutes les requêtes dans le groupe de travail, vous pouvez choisir de configurer les paramètres du groupe de travail. Ils comprennent un emplacement Simple Storage Service (Amazon S3) pour le stockage des résultats des requêtes, le propriétaire du compartiment attendu, le chiffrement et le contrôle des objets écrits dans le compartiment des résultats des requêtes. Vous pouvez

	également imposer des paramètres de groupe de travail. Pour plus d'informations, consultez Paramètres du groupe de travail.
Imposer des contraintes de coûts.	Vous pouvez définir deux types de contraintes de coûts pour les requêtes d'un groupe de travail : • Limite par requête est un seuil pour la quantité de données analysées pour chaque requête. Athena annule les requêtes lorsqu'elles dépassent le seuil spécifié. La limite s'applique à chaque requête en cours d'exécution au sein d'un groupe de travail. Vous pouvez définir une seule limite par requête et la mettre à jour si nécessaire. • La Per-workgroup limit (Limite par groupe de travail) est un seuil que vous pouvez définir pour chaque groupe de travail pour la quantité de données analysées par requêtes dans le groupe de travail. Le dépassement d'un seuil active une alarme Amazon SNS qui déclenche une action de votre choix, par exemple l'envoi d'un e-mail à un utilisateur spécifié. Vous pouvez définir plusieurs limites par groupe de travail pour chaque groupe de travail. Pour obtenir des instructions complètes, consultez Définition des limites pour le contrôle d'utilisation des données.
Suivez les métriques liées aux requêtes pour toutes les requêtes de groupe de travail dans. CloudWatch	Pour chaque requête exécutée dans un groupe de travail, si vous configurez le groupe de travail pour qu'il publie des métriques, Athena les publie sur. CloudWatch Vous pouvez consulter des métriques de requête pour chacun de vos groupes de travail dans la console Athena. Dans CloudWatch, vous pouvez créer des tableaux de bord personnalisés et définir des seuils et des alarmes pour ces mesures.

Fonctionnement des groupes de travail

Les groupes de travail dans Athena présentent les caractéristiques suivantes :

- Par défaut, chaque compte possède un groupe de travail principal et les autorisations par défaut autorisent tous les utilisateurs authentifiés à accéder à ce groupe de travail. Le groupe de travail principal ne peut pas être supprimé.
- Chaque groupe de travail que vous créez indique les requêtes enregistrées et l'historique des requêtes pour les requêtes exécutées uniquement, et non pour toutes les requêtes dans le compte. Cela sépare vos requêtes des autres au sein d'un compte et vous permet de localiser plus efficacement vos propres requêtes enregistrées et les requêtes de l'historique.
- La désactivation d'un groupe de travail empêche l'exécution de requêtes au sein de celui-ci, jusqu'à ce que vous l'activiez. Les requêtes envoyées vers un groupe de travail désactivé échouent, jusqu'à ce que vous l'activiez à nouveau.
- Si vous disposez des autorisation, vous pouvez supprimer un groupe de travail vide, mais aussi un groupe de travail qui contient des requêtes enregistrées. Dans ce cas, avant de supprimer un groupe de travail, Athena vous avertit que les requêtes enregistrées sont supprimées. Avant de supprimer un groupe de travail auquel d'autres utilisateurs ont accès, assurez-vous que les utilisateurs de ce groupe ont accès à d'autres groupes de travail dans lesquels ils peuvent continuer à exécuter des requêtes.
- Vous pouvez configurer des paramètres à l'échelle du groupe de travail et appliquer leur utilisation par toutes les requêtes qui s'exécutent dans un groupe de travail. Les paramètres incluent l'emplacement des résultats de la requête dans Simple Storage Service (Amazon S3), le propriétaire du compartiment attendu, le chiffrement et le contrôle des objets écrits dans le compartiment de résultats de requête.

> **Important**
>
> Lorsque vous appliquez des paramètres à l'échelle du groupe de travail, toutes les requêtes qui s'exécutent dans ce groupe de travail utilisent les paramètres du groupe de travail. Cela se produit même si leurs paramètres côté client diffèrent des paramètres du groupe de travail. Pour plus d'informations, veuillez consulter Les paramètres du groupe de travail remplacent les paramètres côté client.

Limitations pour les groupes de travail

- Vous pouvez créer jusqu'à 1 000 groupes de travail par région dans votre compte.
- Le groupe de travail principal ne peut pas être supprimé.

- Vous pouvez ouvrir jusqu'à dix onglets de requête au sein de chaque groupe de travail. Lorsque vous basculez entre des groupes de travail, vos onglets de requête restent ouverts pour un maximum de trois groupes de travail.

Configuration de groupes de travail

Configurer des groupes de travail implique de les créer et d'établir des autorisations pour leur utilisation. Tout d'abord, décidez de quels groupes de travail votre organisation a besoin, et créez-les. Ensuite, configurez les politiques de groupe de travail IAM qui contrôlent l'accès des utilisateurs et les actions sur une ressource `workgroup`. Les utilisateurs ayant accès à ces groupes de travail peuvent désormais y exécuter des requêtes.

> **Note**
>
> Utilisez ces tâches pour configurer des groupes de travail lorsque vous commencez à les utiliser pour la première fois. Si votre compte Athena utilise déjà des groupes de travail, chaque utilisateur du compte doit disposer des autorisations nécessaires pour exécuter des requêtes dans un ou plusieurs groupes de travail dans le compte. Avant d'exécuter des requêtes, vérifiez votre politique IAM pour voir à quels groupes de travail vous pouvez accéder, ajustez votre politique si nécessaire et basculez vers le groupe de travail que vous prévoyez d'utiliser.

Par défaut, si vous n'avez pas créé de groupes de travail, toutes les requêtes de votre compte s'exécutent dans le groupe de travail principal.

Athena affiche le groupe de travail actuel dans l'option Workgroup (Groupe de travail) dans le coin supérieur droit de la console. Vous pouvez utiliser cette option pour passer d'un groupe de travail à un autre. Lorsque vous exécutez des requêtes, elles s'exécutent dans le groupe de travail existant. Vous pouvez exécuter des requêtes dans le contexte d'un groupe de travail dans la console, à l'aide d'opérations d'API, par l'interface de ligne de commande ou en utilisant une application cliente via le pilote JDBC ou ODBC. Lorsque vous avez accès à un groupe de travail, vous pouvez consulter ses paramètres, ses métriques et ses limites de contrôle d'utilisation des données. Des autorisations supplémentaires vous permettent de modifier les paramètres et les limites de contrôle d'utilisation des données.

Pour configurer des groupes de travail

1. Définir des groupes de travail à créer. Par exemple, vous pouvez décider de ce qui suit :

   - Qui peut exécuter des requêtes dans chaque groupe de travail, et à qui appartient la configuration du groupe de travail. Cela détermine les politiques IAM que vous créez. Pour plus d'informations, consultez Politiques IAM pour l'accès aux groupes de travail.
   - Quels emplacements utiliser dans Simple Storage Service (Amazon S3) pour les résultats des requêtes exécutées dans chaque groupe de travail. Il faut qu'un emplacement existe dans Simple Storage Service (Amazon S3) avant de pouvoir le spécifier pour les résultats de la requête du groupe de travail. Tous les utilisateurs qui utilisent un groupe de travail doivent avoir accès à cet emplacement. Pour plus d'informations, consultez Paramètres du groupe de travail.
   - Indique si le propriétaire du compartiment de résultats de requête Simple Storage Service (Amazon S3) a un contrôle total sur les nouveaux objets écrits dans le compartiment. Par exemple, si l'emplacement de résultat de votre requête appartient à un autre compte, vous pouvez accorder la propriété et le contrôle total des résultats de vos requêtes à l'autre compte. Pour plus d'informations, consultez AclConfiguration.
   - Spécifiez l'ID du compartiment d'emplacement de sortie Compte AWS que vous pensez être le propriétaire. Il s'agit d'une mesure de sécurité optionnelle supplémentaire. Si l'ID de compte du propriétaire du compartiment ne correspond pas à celui que vous spécifiez ici, les tentatives de sortie vers le compartiment échoueront. Pour plus d'informations, consultez Vérification de la propriété du compartiment avec la condition de propriétaire du compartiment dans le Guide de l'utilisateur Simple Storage Service (Amazon S3). Ce paramètre ne s'applique pas aux instructions CTAS, INSERT INTO ou UNLOAD.
   - Quels paramètres de chiffrement sont obligatoires, et quels groupes de travail ont des requêtes qui doivent être chiffrées. Nous vous recommandons de créer des groupes de travail distincts pour les requêtes chiffrées et non chiffrées. De cette façon, vous pouvez appliquer le chiffrement pour un groupe de travail qui s'applique à toutes les requêtes qui s'y exécutent. Pour plus d'informations, consultez Chiffrement des résultats des requêtes Athena stockées dans Simple Storage Service (Amazon S3).
2. Créer des groupes de travail selon vos besoins, et leur ajouter des identifications. Pour les étapes, consultez Créer un groupe de travail.
3. Création de politiques IAM pour vos utilisateurs, groupes ou rôles afin d'activer leur accès aux groupes de travail. Les politiques établissent l'appartenance au groupe de travail et l'accès aux actions sur une ressource `workgroup`. Pour obtenir des instructions complètes, consultez

Politiques IAM pour l'accès aux groupes de travail. Pour voir des exemples les politiques JSON, consultez Accès aux groupes de travail et identifications.

4. Définir les paramètres du groupe de travail Spécifiez un emplacement dans Simple Storage Service (Amazon S3) pour les résultats de la requête et, en option, spécifiez le propriétaire du compartiment attendu, les paramètres de chiffrement et le contrôle des objets écrits dans le compartiment des résultats de la requête. Vous pouvez imposer des paramètres de groupe de travail. Pour en savoir plus, consultez Paramètres de groupe de travail.

   > **Important**
   >
   > Si vous annulez les paramètres côté client, Athena utilisera les paramètres du groupe de travail. Ce paramètre affecte les requêtes que vous exécutez dans la console, en utilisant les pilotes, l'interface de ligne de commande ou les opérations d'API.
   > Pendant que les requêtes continuent de s'exécuter, l'automatisation basée sur la disponibilité des résultats dans un certain compartiment Simple Storage Service (Amazon S3) peut s'interrompre. Nous vous recommandons d'informer vos utilisateurs avant tout remplacement. Une fois les paramètres du groupe de travail définis pour en remplacer d'autres, vous pouvez omettre de spécifier les paramètres côté client dans les pilotes ou l'API.

5. Aviser les utilisateurs des groupes de travail à utiliser pour les requêtes en cours d'exécution. Envoyez un e-mail pour informer les utilisateurs de votre compte des noms de groupe de travail qu'ils peuvent utiliser, des politiques IAM requises et des paramètres du groupe de travail.

6. Configurer les limites de contrôle des coûts pour les requêtes et les groupes de travail, également connues sous le nom de limites de contrôle d'utilisation des données. Pour recevoir une notification lors d'une utilisation hors limites, créez une rubrique Amazon SNS et configurez des abonnements. Pour des étapes détaillées, consultez Définition des limites pour le contrôle d'utilisation des données et Démarrage avec Amazon SNS dans le Guide du développeur Amazon Simple Notification Service.

7. Basculez vers le groupe de travail, pour exécuter des requêtes. Pour exécuter des requêtes, basculez vers le groupe de travail approprié. Pour obtenir des instructions complètes, consultez the section called “Spécifier un groupe de travail dans lequel exécuter des requêtes”.

Politiques IAM pour l'accès aux groupes de travail

Pour contrôler l'accès aux groupes de travail, utilisez des autorisations IAM au niveau des ressources ou des politiques IAM basées sur l'identité. Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

> **Note**
>
> Pour accéder aux groupes de travail compatibles avec la propagation d'identités fiables, les utilisateurs d'IAM Identity Center doivent être affectés à `IdentityCenterApplicationArn` ce qui est renvoyé par la réponse de l'action de l'API GetWorkGroupAthena.

La procédure suivante est spécifique à Athena.

Pour des informations spécifiques à IAM, consultez les liens répertoriés à la fin de cette section. Pour obtenir des informations sur les exemples de politiques de groupes de travail JSON, consultez Exemples de politiques de groupe de travail.

Utilisation de l'éditeur visuel de la console IAM pour créer une politique de groupe de travail

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation à gauche, choisissez politiques, puis Créer une politique.
3. Dans l'onglet Visual editor (Éditeur visuel), sélectionnez Choose a service (Choisir un service). Choisissez ensuite Athena pour l'ajouter à la politique.
4. Choisissez Sélectionner des actions, puis choisissez les actions à ajouter à la politique. L'éditeur visuel affiche les actions disponibles dans Athena. Pour plus d'informations, consultez la rubrique Actions, ressources et clés de condition pour Amazon Athena dans la section Référence de l'autorisation de service.
5. Choisissez Add actions (Ajouter des actions) pour entrer une action spécifique ou utilisez des caractères génériques (*) pour spécifier plusieurs actions.

   Par défaut, la politique que vous créez autorise les actions que vous choisissez. Si vous avez choisi une ou plusieurs actions qui prennent en charge les autorisations au niveau des

ressources pour la ressource `workgroup` dans Athena, l'éditeur visuel affiche la ressource `workgroup`.

6. Choisissez Ressources pour spécifier les groupes de travail spécifiques à votre politique. Pour des exemples de politiques de groupe de travail JSON, consultez Exemples de politiques de groupe de travail.
7. Spécifiez la ressource du `workgroup` comme suit :

```
arn:aws:athena:<region>:<user-account>:workgroup/<workgroup-name>
```

8. Choisissez Review policy (Examiner une politique), puis saisissez un Name (Nom) et une Description (facultatif) pour la politique que vous êtes en train de créer. Passez en revue le résumé de politique afin de vous assurer que les autorisations nécessaires vous ont été accordées.
9. Choisissez Create policy (Créer une politique) pour enregistrer votre nouvelle politique.
10. Attachez cette politique basée sur l'identité à un utilisateur, un groupe ou un rôle.

Pour plus d'informations, consultez les rubriques suivantes dans la Référence de l'autorisation de service et le Guide de l'utilisateur IAM :

- Actions, ressources et clés de condition pour Amazon Athena
- Création de politiques avec l'éditeur visuel
- Ajout et suppression de politiques IAM
- Contrôle de l'accès aux ressources

Pour des exemples de politiques de groupe de travail JSON, consultez Exemples de politiques de groupe de travail.

Pour obtenir une liste complète d'actions Amazon Athena, consultez les noms d'action API dans la rubrique Référence d'API Amazon Athena.

Exemples de politiques de groupe de travail

Cette section inclut des exemples de politiques que vous pouvez utiliser pour activer plusieurs actions sur des groupes de travail. Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Un groupe de travail est une ressource IAM gérée par Athena. Ainsi, si la politique de votre groupe de travail utilise des actions prenant `workgroup` comme entrée, vous devez préciser l'ARN du groupe de travail comme suit :

```
"Resource": [arn:aws:athena:<region>:<user-account>:workgroup/<workgroup-name>]
```

Où *<workgroup-name>* est le nom de votre groupe de travail. Par exemple, pour le groupe de travail nommé `test_workgroup`, spécifiez-le en tant que ressource comme suit :

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"]
```

Pour obtenir une liste complète d'actions Amazon Athena, consultez les noms d'action API dans la rubrique Référence d'API Amazon Athena. Pour plus d'informations sur les politiques IAM, consultez la rubrique Création de politiques avec l'éditeur visuel du Guide de l'utilisateur IAM. Pour plus d'informations sur la création de politiques IAM pour les groupes de travail, voir Politiques IAM pour l'accès aux groupes de travail.

- Example policy for full access to all workgroups
- Example policy for full access to a specified workgroup
- Example policy for running queries in a specified workgroup
- Example policy for running queries in the primary workgroup
- Example policy for management operations on a specified workgroup
- Example policy for listing workgroups
- Example policy for running and stopping queries in a specific workgroup
- Example policy for working with named queries in a specific workgroup
- Example policy for working with Spark notebooks

Example Exemple de politique pour un accès complet à tous les groupes de travail

La politique suivante permet un accès complet à toutes les ressources de groupes de travail susceptibles d'exister dans le compte. Nous vous recommandons d'utiliser cette politique pour ces utilisateurs dans votre compte qui doivent administrer et gérer les groupes de travail pour tous les autres utilisateurs.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Example Exemple de politique pour un accès complet à un groupe de travail spécifié

La politique suivante autorise un accès complet à une seule ressource de groupe de travail spécifique nommée `workgroupA`. Vous pouvez utiliser cette politique pour les utilisateurs ayant un contrôle total sur un groupe de travail en particulier.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListEngineVersions",
                "athena:ListWorkGroups",
                "athena:ListDataCatalogs",
                "athena:ListDatabases",
                "athena:GetDatabase",
                "athena:ListTableMetadata",
                "athena:GetTableMetadata"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:BatchGetQueryExecution",
                "athena:GetQueryExecution",
                "athena:ListQueryExecutions",
                "athena:StartQueryExecution",
                "athena:StopQueryExecution",
```

```
                "athena:GetQueryResults",
                "athena:GetQueryResultsStream",
                "athena:CreateNamedQuery",
                "athena:GetNamedQuery",
                "athena:BatchGetNamedQuery",
                "athena:ListNamedQueries",
                "athena:DeleteNamedQuery",
                "athena:CreatePreparedStatement",
                "athena:GetPreparedStatement",
                "athena:ListPreparedStatements",
                "athena:UpdatePreparedStatement",
                "athena:DeletePreparedStatement"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:DeleteWorkGroup",
                "athena:UpdateWorkGroup",
                "athena:GetWorkGroup",
                "athena:CreateWorkGroup"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA"
            ]
        }
    ]
}
```

Example Exemple de politique pour des requêtes en cours d'exécution dans un groupe de travail spécifié

Dans la politique suivante, un utilisateur est autorisé à exécuter des requêtes dans le `workgroupA` spécifié, et de les afficher. L'utilisateur n'est pas autorisé à effectuer des tâches de gestion pour le groupe de travail lui-même, telles que le mettre à jour ou le supprimer.

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

```
            "Effect": "Allow",
            "Action": [
                "athena:ListEngineVersions",
                "athena:ListWorkGroups",
                "athena:ListDataCatalogs",
                "athena:ListDatabases",
                "athena:GetDatabase",
                "athena:ListTableMetadata",
                "athena:GetTableMetadata"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:GetWorkGroup",
                "athena:BatchGetQueryExecution",
                "athena:GetQueryExecution",
                "athena:ListQueryExecutions",
                "athena:StartQueryExecution",
                "athena:StopQueryExecution",
                "athena:GetQueryResults",
                "athena:GetQueryResultsStream",
                "athena:CreateNamedQuery",
                "athena:GetNamedQuery",
                "athena:BatchGetNamedQuery",
                "athena:ListNamedQueries",
                "athena:DeleteNamedQuery",
                "athena:CreatePreparedStatement",
                "athena:GetPreparedStatement",
                "athena:ListPreparedStatements",
                "athena:UpdatePreparedStatement",
                "athena:DeletePreparedStatement"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA"
            ]
        }
    ]
}
```

Example Exemple de politique pour des requêtes en cours d'exécution dans le groupe de travail principal

Vous pouvez modifier l'exemple précédent pour permettre à un utilisateur particulier d'exécuter des requêtes dans le groupe de travail principal.

> **Note**
>
> Nous vous recommandons d'ajouter la ressource du groupe de travail principal pour tous les utilisateurs qui sont autrement configurés pour exécuter des requêtes dans leurs groupes de travail désignés. L'ajout de cette ressource aux politiques d'utilisation de leur groupe de travail est utile au cas où le groupe de travail désigné serait supprimé ou désactivé. Dans ce cas, ils peuvent continuer à exécuter des requêtes dans le groupe de travail principal.

Pour permettre aux utilisateurs de votre compte d'exécuter des requêtes dans le groupe de travail principal, ajoutez une ligne contenant l'ARN du groupe de travail principal à la section des ressources de Example policy for running queries in a specified workgroup, comme dans l'exemple ci-dessous.

```
arn:aws:athena:us-east-1:123456789012:workgroup/primary"
```

Example Exemple de politique pour les opérations de gestion sur un groupe de travail spécifié

Dans la politique suivante, un utilisateur est autorisé à créer, supprimer, obtenir des détails, et mettre à jour un groupe de travail `test_workgroup`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListEngineVersions"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateWorkGroup",
                "athena:GetWorkGroup",
```

```
                "athena:DeleteWorkGroup",
                "athena:UpdateWorkGroup"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"
            ]
        }
    ]
}
```

Example Exemple de politique pour répertorier des groupes de travail

La politique suivante autorise tous les utilisateurs à répertorier tous les groupes de travail :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListWorkGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Exemple de politique pour exécuter et arrêter des requêtes dans un groupe de travail spécifié

Dans cette politique, un utilisateur est autorisé à exécuter des requêtes dans le groupe de travail :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:StartQueryExecution",
                "athena:StopQueryExecution"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"
```

```
            ]
        }
    ]
}
```

Example Exemple de politique à utiliser avec des requêtes nommées dans un groupe de travail spécifié

Dans la politique suivante, un utilisateur dispose des autorisations pour créer, supprimer et obtenir des informations sur les requêtes nommées dans le groupe de travail spécifié :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateNamedQuery",
                "athena:GetNamedQuery",
                "athena:DeleteNamedQuery"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"
     ]
        }
    ]
}
```

Example Exemple de politique pour utiliser des blocs-notes Spark dans Athena

Utilisez une politique telle que la suivante pour utiliser les blocs-notes Spark dans Athena.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreatingWorkGroupWithDefaults",
            "Action": [
                "athena:CreateWorkGroup",
                "s3:CreateBucket",
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
```

```
                "s3:GetBucketLocation",
                "athena:ImportNotebook"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/Demo*",
                "arn:aws:s3:::123456789012-us-east-1-athena-results-bucket-*",
                "arn:aws:iam::123456789012:role/service-role/
AWSAthenaSparkExecutionRole-*",
                "arn:aws:iam::123456789012:policy/service-role/
AWSAthenaSparkRolePolicy-*"
            ]
        },
        {
            "Sid": "AllowRunningCalculations",
            "Action": [
                "athena:ListWorkGroups",
                "athena:GetWorkGroup",
                "athena:StartSession",
                "athena:CreateNotebook",
                "athena:ListNotebookMetadata",
                "athena:ListNotebookSessions",
                "athena:GetSessionStatus",
                "athena:GetSession",
                "athena:GetNotebookMetadata",
                "athena:CreatePresignedNotebookUrl"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:athena:us-east-1:123456789012:workgroup/Demo*"
        },
        {
            "Sid": "AllowListWorkGroupAndEngineVersions",
            "Action": [
                "athena:ListWorkGroups",
                "athena:ListEngineVersions"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Paramètres du groupe de travail

Chaque groupe de travail a les paramètres suivants :

- Un nom unique. Il peut contenir entre 1 et 128 caractères, y compris des caractères alphanumériques, des tirets et des traits de soulignement. Une fois le groupe de travail créé, vous ne pouvez plus modifier son nom. Toutefois, vous pouvez créer un nouveau groupe de travail avec les mêmes paramètres et un autre nom.
- Des paramètres qui s'appliquent à toutes les requêtes en cours d'exécution dans le groupe de travail. Il s'agit des sections suivantes :
  - Emplacement dans Simple Storage Service (Amazon S3) pour stocker les résultats des requêtes pour toutes les requêtes qui s'exécutent dans ce groupe de travail. Cet emplacement doit exister avant de pouvoir le spécifier pour le groupe de travail quand vous le créez. Pour plus d'informations sur la création d'un compartiment Amazon S3, consultez Création d'un compartiment.
  - Contrôle du propriétaire du compartiment des résultats de la requête : indique si le propriétaire du compartiment de résultats de requête Simple Storage Service (Amazon S3) a un contrôle total sur les nouveaux objets écrits dans le compartiment. Par exemple, si l'emplacement de résultat de votre requête appartient à un autre compte, vous pouvez accorder la propriété et le contrôle total des résultats de vos requêtes à l'autre compte.
  - Propriétaire attendu du bucket : ID du bucket Compte AWS que vous pensez être le propriétaire du bucket des résultats de la requête. Il s'agit d'une mesure de sécurité supplémentaire. Si l'ID de compte du propriétaire du compartiment ne correspond pas à celui que vous spécifiez ici, les tentatives de sortie vers le compartiment échoueront. Pour obtenir des informations détaillées, consultez Vérification de la propriété du compartiment avec la condition de propriétaire du compartiment dans le Guide de l'utilisateur Simple Storage Service (Amazon S3).

    > **Note**
    >
    > Le paramètre de propriétaire du compartiment attendu s'applique uniquement à l'emplacement de sortie Simple Storage Service (Amazon S3) que vous spécifiez pour les résultats de la requête Athena. Il ne s'applique pas aux autres emplacements Simple Storage Service (Amazon S3) tels que les emplacements de source de données dans des compartiments Simple Storage Service (Amazon S3) externes, des emplacements de table de destination `CTAS` et `INSERT INTO`, des emplacements de sortie d'instruction

UNLOAD, des opérations de déversement de compartiments pour les requêtes fédérées, ou des requêtes SELECT exécutées sur une table d'un autre compte.

- Un paramètre de chiffrement, si vous utilisez le chiffrement pour toutes les requêtes des groupes de travail. Vous pouvez uniquement chiffrer toutes les requêtes dans un groupe de travail, pas seulement certaines d'entre elles. Il est préférable de créer des groupes de travail distincts qui contiennent les requêtes, qu'elles soient chiffrées ou non.

En outre, votre groupe de travail peut remplacer les paramètres côté client. Avant le lancement des groupes de travail, vous pouviez spécifier l'emplacement des résultats et les options de chiffrement en tant que paramètres dans le pilote JDBC ou ODBC, ou dans l'onglet Properties (Propriétés) de la console Athena. Ces paramètres peuvent également être spécifiés directement via les opérations d'API. Ces paramètres sont appelés « paramètres côté client ». Avec des groupes de travail, vous pouvez configurer ces paramètres au niveau du groupe de travail pour contrôler les options disponibles au niveau du client. L'application des paramètres au niveau du groupe de travail évite également aux utilisateurs d'avoir à configurer individuellement leurs paramètres côté client. Si vous sélectionnez l'option Remplacer les paramètres côté client pour le groupe de travail, les requêtes utilisent les paramètres du groupe de travail et ignorent les paramètres côté client.

Si Override Client-Side Setting (Remplacer les paramètres côté client) est sélectionné, l'utilisateur est informé sur la console que ses paramètres ont été modifiés. Si les paramètres du groupe de travail sont appliqués de cette façon, les utilisateurs peuvent omettre les paramètres côté client correspondants. Ensuite, les requêtes exécutées dans la console utilisent les paramètres du groupe de travail même si des paramètres côté client sont présents. En outre, lorsque les requêtes du groupe de travail sont exécutées via des opérations d'API ou des AWS CLI pilotes JDBC ou ODBC, les paramètres côté client tels que l'emplacement des résultats des requêtes et le chiffrement sont remplacés par les paramètres du groupe de travail. Pour consulter les paramètres du groupe de travail, affichez les détails du groupe de travail.

Vous pouvez également définir des limites de requête pour des requêtes dans les groupes de travail.

Les paramètres du groupe de travail remplacent les paramètres côté client

Les boîtes de dialogue Create workgroup (Créer un groupe de travail) et Edit workgroup (Modifier un groupe de travail) ont un champ intitulé Override Client-Side Setting (Remplacer les paramètres côté client). Ce champ n'est pas sélectionné par défaut. En fonction de votre choix, Athena effectue les actions suivantes :

- Si Remplacer les paramètres côté client n'est pas sélectionné, les paramètres du groupe de travail ne sont pas appliqués au niveau du client. Lorsque l'option Remplacer les paramètres côté client n'est pas sélectionnée pour le groupe de travail, Athena utilise les paramètres côté client pour toutes les requêtes exécutées dans le groupe de travail, y compris l'emplacement des résultats des requêtes, le propriétaire du compartiment attendu, le chiffrement et le contrôle des objets écrits dans le compartiment des résultats des requêtes. Chaque utilisateur peut spécifier ses propres paramètres dans le menu Paramètres sur la console. Si les paramètres côté client ne sont pas définis, les paramètres à l'échelle du groupe de travail s'appliquent. Si vous utilisez les AWS CLI actions d'API ou les pilotes JDBC et ODBC pour exécuter des requêtes dans un groupe de travail qui ne remplacent pas les paramètres côté client, vos requêtes utilisent les paramètres que vous spécifiez dans vos requêtes.
- Si Remplacer les paramètres côté client est sélectionné, les paramètres du groupe de travail sont appliqués au niveau du groupe de travail pour tous les clients du groupe de travail. Lorsque l'option Remplacer les paramètres côté client est sélectionnée pour le groupe de travail, Athena utilise les paramètres du groupe de travail pour toutes les requêtes exécutées dans le groupe de travail, y compris l'emplacement des résultats des requêtes, le propriétaire du compartiment attendu, le chiffrement et le contrôle des objets écrits dans le compartiment des résultats des requêtes. Les paramètres du groupe de travail remplacent tous les paramètres côté client que vous spécifiez pour une requête lorsque vous utilisez la console, les actions d'API ou les pilotes JDBC ou ODBC.

  Si vous remplacez les paramètres côté client, la prochaine fois que vous ou un utilisateur du groupe de travail ouvrirez la console Athena, Athena vous informera que les requêtes dans le groupe de travail utilisent les paramètres du groupe de travail et vous invitera à confirmer cette modification.

  > **Important**
  >
  > Si vous utilisez des actions d'API AWS CLI, les pilotes JDBC ou ODBC pour exécuter des requêtes dans un groupe de travail qui remplacent les paramètres côté client, veillez à omettre les paramètres côté client dans vos requêtes ou à les mettre à jour pour qu'ils correspondent aux paramètres du groupe de travail. Si vous spécifiez des paramètres côté client dans vos requêtes mais que vous les exécutez dans un groupe de travail qui remplace les paramètres, les requêtes seront exécutées mais en utilisant les paramètres du groupe de travail. Pour plus d'informations sur l'affichage des paramètres d'un groupe de travail, consultez Afficher les détails du groupe de travail.

Gestion des groupes de travail

Dans https://console.aws.amazon.com/athena/, vous pouvez effectuer les tâches suivantes :

Instruction	Description
Créer un groupe de travail	Créer un nouveau groupe de travail.
Modifier un groupe de travail	Modifier un groupe de travail et ses paramètres. Vous ne pouvez pas modifier le nom d'un groupe de travail, mais vous pouvez créer un nouveau groupe de travail avec les mêmes paramètres et un autre nom.
Afficher les détails du groupe de travail	Affichez les détails du groupe de travail, tels que son nom, sa descripti on, les limites d'utilisation des données, l'emplacement des résultats de la requête, le propriétaire prévu du compartiment des résultats de la requête, le chiffrement et le contrôle des objets écrits dans le compartiment des résultats de la requête. Vous pouvez également vérifier si ce groupe de travail applique ses paramètres, si Override Client-Side Setting (Remplacer les paramètres côté client) est sélectionné.
Supprimer un groupe de travail	Supprimer un groupe de travail. Si vous supprimez un groupe de travail, l'historique des requêtes, les requêtes enregistrées, les paramètres du groupe de travail et les contrôles de limite des données par requête sont supprimés. Les contrôles de limite de données à l'échelle du groupe de travail restent actifs CloudWatch et vous pouvez les supprimer individuellement. Le groupe de travail principal ne peut pas être supprimé.
Changer de groupe de travail	Basculer entre des groupes de travail auxquels vous avez accès.
Copier une requête enregistrée entre les groupes de travail	Copier une requête enregistrée entre les groupes de travail. Vous pouvez procéder ainsi si, par exemple, vous avez créé une requête dans un groupe de travail de prévisualisation et que vous souhaitez la rendre disponible dans un groupe de travail sans prévisualisation.

Instruction	Description
Activer et désactiver un groupe de travail	Activer ou désactiver un groupe de travail. Lorsqu'un groupe de travail est désactivé, ses utilisateurs ne peuvent pas exécuter de requêtes ou créer de nouvelles requêtes nommées. Si vous y avez accès, vous pouvez toujours afficher les métriques, les contrôles de limite d'utilisation des données, les paramètres du groupe de travail, l'historique des requêtes et les requêtes enregistrées.
Spécifier un groupe de travail dans lequel exécuter des requêtes	Avant de pouvoir exécuter des requêtes, vous devez indiquer à Athena le groupe de travail à utiliser. Vous devez avoir les autorisations nécessaires au groupe de travail.
Création d'un groupe de travail Athena qui utilise l'authentification IAM Identity Center	Pour utiliser les identités IAM Identity Center avec Athena, vous devez créer un groupe de travail compatible avec IAM Identity Center. Après avoir créé le groupe de travail, vous pouvez utiliser la console ou l'API IAM Identity Center pour attribuer des utilisateurs ou des groupes IAM Identity Center au groupe de travail.

Créer un groupe de travail

Créer un groupe de travail nécessite des autorisations pour les actions d'API `CreateWorkgroup`. Consultez Accès aux groupes de travail et identifications et Politiques IAM pour l'accès aux groupes de travail. Si vous ajoutez des identifications, vous devez également ajouter des autorisations à `TagResource`. veuillez consulter Exemples de politique d'identification pour les groupes de travail.

Pour créer un groupe de travail dans la console

1. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

2. Dans le panneau de navigation de la console Athena, choisissez Workgroups (Groupes de travail).
3. Sur la page Workgroups (Groupes de travail), choisissez Create workgroup (Créer un groupe de travail).
4. Sur la page Create workgroup (Créer un groupe de travail), remplissez les champs comme suit :

Champ	Description
Nom du groupe de travail	Obligatoire. Saisissez un nom unique pour votre groupe de travail. Utilisez 1 à 128 caractères. (A-Z,a-z,0-9,_,-,.). Ce nom ne peut pas être modifié.
Description	Facultatif. Saisissez une description pour votre groupe de travail. Elle peut contenir jusqu'à 1 024 caractères.
Choisissez le type de moteur	Choisissez Athena SQL si vous souhaitez exécuter des requêtes SQL ad hoc sur les données d'Amazon S3 ou utilisez un connecteur prédéfini de source de données pour exécuter des requêtes fédérées sur une variété de sources de données externes à Amazon S3. Vous pouvez exécuter des requêtes à l'aide de l'éditeur de requêtes Athena, de l'interface AWS CLI ou des API Athena.

Champ	Description
	Choisissez Apache Spark si vous souhaitez créer, modifier et exécuter des applications de bloc-notes Jupyter à l'aide de Python et d'Apache Spark. Les blocs-notes Jupyter contiennent une liste de cellules qui peuvent inclure du code, du texte standard, du texte au format Markdown, des mathématiques, des graphique s et du contenu multimédia enrichi. Les cellules sont exécutées dans l'ordre sous forme de calculs dans une session de bloc-note s interactive dans Athena. Pour plus d'informations sur la création et la configuration d'un groupe de travail compatible avec Spark, voir Création d'un groupe de travail compatible avec Spark dans Athena. Après la création d'un groupe de travail, son moteur d'analyse peut être mis à niveau (par exemple, de la version 2 du moteur Athena à la version 3 du moteur Athena), mais son type de moteur ne peut pas être modifié. Par exemple, un groupe de travail du moteur Athena version 3 ne peut pas être remplacé par un groupe de travail PySpark du moteur version 3.
Mise à jour du moteur de requête	Choisissez la façon dont vous souhaitez mettre à jour votre groupe de travail lorsqu'une nouvelle version du moteur Athena est publiée. Vous pouvez laisser Athena décider du moment de la mise à jour de votre groupe de travail ou choisir manuellement une version du moteur. Pour plus d’informations, consultez Gestion des versions du moteur Athena.
Mode d’authentification	Choisissez AWS Identity and Access Management (IAM) pour utiliser l’authentification ou la fédération IAM pour le groupe de travail. Choisissez IAM Identity Center si vous souhaitez prendre en charge les identités du personnel telles que les utilisateurs et les groupes provenant de fournisseurs d’identité SAML 2.0 comme Microsoft Active Directory. Pour plus d’informations, consultez la rubrique Trusted identity propagation across applications dans le Guide de l’utilisateur AWS IAM Identity Center .

Champ	Description
Fonction du service pour l’accès à IAM Identity Center	Athena a besoin des autorisations IAM pour accéder à IAM Identity Center en votre nom. Pour plus d’informations sur les fonctions du service IAM, consultez Création d’un rôle pour la délégation d’autorisations à un service AWS dans le Guide de l’utilisateur IAM.
Emplacement des résultats de requête	Facultatif. Saisissez un chemin vers un compartiment Simple Storage Service (Amazon S3) ou un préfixe. Ce compartiment et ce préfixe doivent exister avant que vous puissiez les spécifier. **Note** Si vous exécutez des requêtes dans la console, spécifier l'emplacement des résultats de requête est facultatif. Si vous ne le spécifiez pas pour le groupe de travail ou dans Paramètres, Athena utilise l'emplacement de résultat de la requête par défaut. Si vous exécutez des requêtes avec l'API ou les pilotes, vous devez spécifier l'emplacement des résultats de la requête au moins à l'un des deux endroits suivants : pour les requêtes individuelles avec OutputLocation, ou pour le groupe de travail, avec WorkGroup Configuration.

Champ	Description
Propriétaire du compartiment attendu	Facultatif. Entrez l'ID du compartiment de localisation de sortie Compte AWS que vous pensez être le propriétaire. Il s'agit d'une mesure de sécurité supplémentaire. Si l'ID de compte du propriétaire du compartiment ne correspond pas à l'ID que vous spécifiez, les tentatives de sortie vers le compartiment échoueront. Pour obtenir des informations détaillées, consultez Vérification de la propriété du compartiment avec la condition de propriétaire du compartiment dans le Guide de l'utilisateur Simple Storage Service (Amazon S3). **Note** Le paramètre de propriétaire du compartiment attendu s'applique uniquement à l'emplacement de sortie Simple Storage Service (Amazon S3) que vous spécifiez pour les résultats de la requête Athena. Il ne s'applique pas aux autres emplacements Simple Storage Service (Amazon S3) tels que les emplacements de source de données dans des compartiments Simple Storage Service (Amazon S3) externes, des emplacements de table de destination `CTAS` et `INSERT INTO`, des emplacements de sortie d'instruction `UNLOAD`, des opérations de déversement de compartiments pour les requêtes fédérées, ou des requêtes `SELECT` exécutées sur une table d'un autre compte.

Champ	Description
Attribuer au propriétaire du compartiment un contrôle total sur les résultats de la requête	Ce champ n'est pas sélectionné par défaut. Si vous la sélectionnez et que les listes de contrôle d'accès (ACL) sont activées pour le compartiment d'emplacement des résultats de la requête, vous accordez un accès de contrôle total aux résultats de la requête au propriétaire du compartiment. Par exemple, si l'emplacement de résultat de votre requête appartient à un autre compte, vous pouvez accorder la propriété et le contrôle total des résultats de vos requêtes à l'autre compte. Si le paramètre S3 Object Ownership du compartiment est défini à Propriétaire du compartiment préféré, le propriétaire du compartiment possède également tous les objets de résultats de requête écrits à partir de ce groupe de travail. Par exemple, si le groupe de travail d'un compte externe active cette option et définit son emplacement de résultat de requête sur le compartiment Simple Storage Service (Amazon S3) de votre compte qui dispose d'un paramètre de S3 Object Ownership dont la valeur est définie à Propriétaire du compartiment préféré, vous possédez et contrôlez complètement les résultats de requête du groupe de travail externe. Sélectionner cette option lorsque le paramètre S3 Object Ownership du compartiment de résultats de requête est défini à Propriétaire du compartiment appliqué n'a aucun effet. Pour de plus amples informations, consultez la rubrique Paramètres de propriété des objets dans le Guide de l'utilisateur Simple Storage Service (Amazon S3).

Champ	Description
Chiffrer les résultats de requête	Facultatif. Chiffrer les résultats stockés dans Simple Storage Service (Amazon S3). Si cette option est sélectionnée, toutes les requêtes dans le groupe de travail sont chiffrées. Si cette option est sélectionnée, vous pouvez sélectionner le Type de chiffrement, la Clé de chiffrement et saisir l'ARN de clé KMS. Si vous n'avez pas la clé, ouvrez la console AWS KMS pour la créer. Pour plus d'informations, consultez Création des clés dans le Guide du développeur AWS Key Management Service .
Définissez ***encryption_type*** comme chiffrement minimal	Facultatif. Sélectionnez cette option pour appliquer un type de chiffrement minimal aux résultats des requêtes pour tous les utilisateurs du groupe de travail. La sélection de cette option affiche un tableau reprenant la hiérarchie des types de chiffrement. Le tableau indique également les types de chiffrement que les utilisateurs des groupes de travail seront autorisés à utiliser lorsque vous spécifiez un type de chiffrement particulier comme minimum. Pour utiliser cette option, l'option Remplacer les paramètres côté client ne doit pas être sélectionnée. Pour plus d’informations, consultez Configuration du chiffrement minimal pour un groupe de travail.
Activer les autorisations d’accès S3	Ce champ est sélectionné par défaut lorsque vous choisissez IAM Identity Center comme mode d’authentification. Si vous activez cette option, elle applique les autorisations basées sur les utilisateurs ou les groupes IAM Identity Center aux emplacements Amazon S3.
Créer un préfixe S3 basé sur l’identité utilisateur	Si vous activez cette option, Athena crée un préfixe Amazon S3 quand elle stocke les résultats des requêtes. Le préfixe est basé sur l’identité de l’utilisateur IAM Identity Center.

Champ	Description
Publier les métriques des requêtes sur CloudWatch	Ce champ est sélectionné par défaut. Publier des métriques de requête dans CloudWatch. veuillez consulter Surveillance des requêtes Athena à l'aide de métriques CloudWatch .
Remplacer les paramètres côté client	Ce champ n'est pas sélectionné par défaut. Si vous sélectionnez cette option, les paramètres du groupe de travail s'appliquent à toutes les requêtes dans le groupe de travail et remplacent les paramètres côté client. Pour plus d'informations, consultez Les paramètres du groupe de travail remplacent les paramètres côté client.
Compartiments S3 de type Paiement par le demandeur	Facultatif. Choisissez Turn on queries on requester pays buckets in Amazon S3 (Activation des requêtes sur les compartiments de type Paiement par le demandeur dans Amazon S3) si les utilisate urs du groupe de travail exécutent des requêtes sur les données stockées dans des compartiments Amazon S3 configurés comme des compartiments de type Paiement par le demandeur. Le compte de l'utilisateur exécutant la requête est facturé pour les frais d'accès aux données et de transfert de données associés à la requête. Pour plus d'informations, consultez la section Compartiments de type Paiement par le demandeur dans le Guide de l'utilisateur d'Amazon Simple Storage Service.
Gestion du contrôle de l'utilisation des données par requête	Facultatif. Définit la limite de la quantité maximale de données qu'une requête est autorisée à analyser. Vous ne pouvez définir qu'une seule limite par requête pour un groupe de travail. La limite s'applique à toutes les requêtes dans le groupe de travail. Si la requête dépasse la limite, elle sera annulée. Pour plus d'informa tions, consultez Définition des limites pour le contrôle d'utilisation des données.

Champ	Description
Workgroup data usage alerts (Alertes d'utilisation des données de groupe de travail	Facultatif. Définissez plusieurs seuils d'alerte lorsque des requêtes exécutées dans ce groupe de travail analysent une quantité de données spécifiée au cours d'une période donnée. Les alertes sont mises en œuvre à l'aide des CloudWatch alarmes Amazon et s'appliquent à toutes les requêtes du groupe de travail. Pour plus d'informations, consultez la section Utilisation des CloudWatch alarmes Amazon dans le guide de CloudWatch l'utilisateur Amazon.
Balises	Facultatif. Ajoutez une ou plusieurs identifications à un groupe de travail. une identification est une étiquette que vous affectez à une ressource d'un groupe de travail Athena. Elle se compose d'une clé et d'une valeur. Utilisez AWS les meilleures pratiques en matière de balisage pour créer un ensemble cohérent de balises et classer les groupes de travail par objectif, propriétaire ou environnement. Vous pouvez également utiliser des identifications dans des politiques IAM et contrôler les coûts de facturation. N'utilisez pas de clés d'identification dupliquées pour le même groupe de travail. Pour plus d'informations, consultez the section called "Étiquetage des ressources".

5. Choisissez Create workgroup (Créer un groupe de travail). Le groupe de travail s'affiche sur la liste de la page Workgroups (Groupes de travail).

Vous pouvez également utiliser l'opération CreateWorkGroupAPI pour créer un groupe de travail.

> **Important**
>
> Après avoir créé des groupes de travail, créez des Politiques IAM pour l'accès aux groupes de travail IAM qui vous permettent d'exécuter des actions liées aux groupes de travail.

Modifier un groupe de travail

Modifier un groupe de travail nécessite les autorisations requises pour les opérations d'API `UpdateWorkgroup`. Consultez Accès aux groupes de travail et identifications et Politiques IAM pour l'accès aux groupes de travail. Si vous ajoutez ou modifiez des identifications, vous devez également

avoir des autorisations pour `TagResource`. veuillez consulter Exemples de politique d'identification pour les groupes de travail.

Pour modifier un groupe de travail dans la console

1. Dans le panneau de navigation de la console Athena, choisissez Workgroups (Groupes de travail).
2. Sur la page Workgroups (Groupes de travail), sélectionnez le bouton du groupe de travail à modifier.
3. Choisissez Actions, Edit (Modifier).
4. Modifiez les champs si nécessaire. Pour obtenir la liste des champs, consultez Créer un groupe de travail. Vous pouvez modifier tous les champs à l'exception du nom du groupe de travail. Si vous devez modifier le nom, créez un autre groupe de travail avec le nouveau nom et les mêmes paramètres.
5. Sélectionnez Enregistrer les modifications. Le groupe de travail mis à jour s'affiche sur la liste de la page Workgroups (Groupes de travail).

Afficher les détails du groupe de travail

Vous pouvez afficher les détails de chaque groupe de travail. Les détails comprennent le nom du groupe de travail, sa description, le fait qu'il soit activé ou désactivé, et les paramètres utilisés pour les requêtes qui s'exécutent dans le groupe de travail, qui comprennent l'emplacement des résultats de la requête, le propriétaire attendu du compartiment, le chiffrement et le contrôle des objets écrits dans le compartiment des résultats de la requête. Si un groupe de travail a des limites d'utilisation des données, elles sont également affichées.

Pour afficher les détails du groupe de travail

1. Dans le panneau de navigation de la console Athena, choisissez Workgroups (Groupes de travail).
2. Sur la page Workgroups (Groupes de travail), choisissez le lien du groupe de travail à consulter. La page Overview Details (Détails de présentation) pour le groupe de travail s'affiche.

Supprimer un groupe de travail

Vous pouvez supprimer un groupe de travail si vous en avez l'autorisation. Le groupe de travail principal ne peut pas être supprimé.

Si vous disposez des autorisations, vous pouvez supprimer un groupe de travail vide à tout moment. Vous pouvez également supprimer un groupe de travail qui contient des requêtes enregistrées. Dans ce cas, avant de procéder à la suppression d'un groupe de travail, Athena vous avertit que les requêtes enregistrées sont supprimées.

Si vous supprimez un groupe de travail pendant que vous vous y trouvez, la console se concentre sur le groupe de travail principal. Si vous y avez accès, vous pouvez exécuter des requêtes et afficher ses paramètres.

Si vous supprimez un groupe de travail, ses paramètres et ses contrôles de limite de données par requête sont supprimés. Les contrôles de limite de données à l'échelle du groupe de travail restent CloudWatch actifs et vous pouvez les supprimer si nécessaire.

> **Important**
>
> Avant de supprimer un groupe de travail, assurez-vous que ses utilisateurs ont également accès à d'autres groupes de travail dans lesquels ils peuvent continuer à exécuter des requêtes. Si les politiques IAM des utilisateurs leur permettaient d'exécuter des requêtes uniquement dans ce groupe de travail et que vous le supprimez, ils n'ont plus l'autorisation d'exécuter des requêtes. Pour plus d'informations, consultez Example policy for running queries in the primary workgroup.

Pour supprimer un groupe de travail dans la console

1. Dans le panneau de navigation de la console Athena, choisissez Workgroups (Groupes de travail).
2. Sur la page Workgroups (Groupes de travail), sélectionnez le bouton du groupe de travail à supprimer.
3. Sélectionnez Actions, Supprimer.
4. Lorsque l'invite de confirmation Delete workgroup (Supprimer le groupe de travail) s'affiche, saisissez le nom du groupe de travail, puis choisissez Delete (Supprimer).

Pour supprimer un groupe de travail avec l'opération d'API, utilisez l'action `DeleteWorkGroup`.

Changer de groupe de travail

Vous pouvez passer d'un groupe de travail à un autre si vous avez les autorisations pour chacun d'entre eux.

Vous pouvez ouvrir jusqu'à dix onglets de requête au sein de chaque groupe de travail. Lorsque vous basculez entre des groupes de travail, vos onglets de requête restent ouverts pour un maximum de trois groupes de travail.

Changement de groupe de travail

1. Dans la console Athena, choisissez l'option Workgroup (Groupe de travail) en haut à droite pour choisir un groupe de travail.
2. Si la boite de dialogue WorkGroup ***workgroup-name*** settings (Paramètres du groupe de travail workgroup-name) s'affiche, choisissez Acknowledge (Confirmer).

L'option Workgroup (Groupe de travail) indique le nom du groupe de travail vers lequel vous avez basculé. Vous pouvez désormais exécuter des requêtes dans ce groupe de travail.

Copier une requête enregistrée entre les groupes de travail

Actuellement, la console Athena ne dispose pas d'une option permettant de copier directement une requête enregistrée d'un groupe de travail à un autre, mais vous pouvez effectuer la même tâche manuellement en utilisant la procédure suivante.

Copier une requête enregistrée entre groupes de travail

1. Dans la console Athena, depuis le groupe de travail à partir duquel vous souhaitez copier la requête, choisissez l'onglet Saved queries (Requêtes enregistrées).
2. Choisissez le lien de la requête enregistrée à copier. Athena ouvre la requête dans l'éditeur de requêtes.
3. Dans l'éditeur de requêtes, sélectionnez le texte de la requête, puis appuyez sur **Ctrl+C** pour le copier.
4. Passez au groupe de travail de destination ou créez un groupe de travail, puis passez-y.
5. Ouvrez un nouvel onglet dans l'éditeur de requêtes, puis appuyez sur **Ctrl+V** pour coller le texte dans le nouvel onglet.
6. Dans l'éditeur de requêtes, choisissez Save as (Enregistrer sous) pour enregistrer la requête dans le groupe de travail de destination.

7. Dans la boîte de dialogue Choose a name (Choisir un nom), saisissez un nom pour la requête et une description facultative.
8. Choisissez Enregistrer.

Activer et désactiver un groupe de travail

Si vous avez les autorisations nécessaires, vous pouvez activer ou désactiver des groupes de travail dans la console, en utilisant les opérations d'API, ou les pilotes JDBC et ODBC.

Pour activer ou désactiver un groupe de travail

1. Dans le panneau de navigation de la console Athena, choisissez Workgroups (Groupes de travail).
2. Sur la page Workgroups (Groupes de travail), choisissez le lien du groupe de travail.
3. Dans le coin supérieur droit, choisissez Enable workgroup (Activer le groupe de travail) ou Disable workgroup (Désactiver le groupe de travail).
4. Lorsque l'invite de confirmation s'affiche, choisissez Enable (Activer) ou Disable (Désactiver). Si vous désactivez un groupe de travail, ses utilisateurs ne peuvent pas y exécuter de requêtes ou y créer de nouvelles requêtes nommées. Si vous activez un groupe de travail, les utilisateurs peuvent l'utiliser pour exécuter des requêtes.

Spécifier un groupe de travail dans lequel exécuter des requêtes

Pour spécifier un groupe de travail à utiliser, vous devez avoir les autorisations nécessaires sur le groupe de travail.

Spécification du groupe de travail à utiliser

1. Assurez-vous que vos autorisations vous permettent d'exécuter des requêtes dans le groupe de travail que vous prévoyez d'utiliser. Pour plus d’informations, consultez the section called “ Politiques IAM pour l'accès aux groupes de travail”.
2. Pour spécifier le groupe de travail, utilisez l'une des options suivantes :
   - Si vous utilisez la console Athena, définissez le groupe de travail en changeant de groupes de travail.

- Si vous utilisez les opérations d'API Athena, spécifiez le nom du groupe de travail dans l'action d'API. Par exemple, vous pouvez définir le nom du groupe de StartQueryExecutiontravail comme suit :

```
StartQueryExecutionRequest startQueryExecutionRequest = new
 StartQueryExecutionRequest()
             .withQueryString(ExampleConstants.ATHENA_SAMPLE_QUERY)
             .withQueryExecutionContext(queryExecutionContext)
             .withWorkGroup(WorkgroupName)
```

- Si vous utilisez le pilote JDBC ou ODBC, définissez le nom du groupe de travail dans la chaîne de connexion à l'aide du paramètre de configuration `Workgroup`. Le pilote transmet le nom du groupe de travail à Athena. Spécifiez le paramètre du groupe de travail dans la chaîne de connexion, comme dans l'exemple suivant :

```
jdbc:awsathena://AwsRegion=<AWSREGION>;UID=<ACCESSKEY>;
PWD=<SECRETKEY>;S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/<athena-
output>-<AWSREGION>/;
Workgroup=<WORKGROUPNAME>;
```

Configuration du chiffrement minimal pour un groupe de travail

En tant qu'administrateur d'un groupe de travail Athena SQL, vous pouvez appliquer un niveau de chiffrement minimal dans Amazon S3 pour tous les résultats de requêtes du groupe de travail. Vous pouvez utiliser cette fonctionnalité pour vous assurer que les résultats des requêtes ne sont jamais stockés dans un compartiment Amazon S3 à l'état non chiffré.

Lorsque les utilisateurs d'un groupe de travail où le chiffrement minimal est activé soumettent une requête, ils peuvent uniquement définir le niveau de chiffrement au niveau minimum que vous avez configuré, ou à un niveau supérieur s'il est disponible. Athena chiffre les résultats de la requête soit au niveau spécifié lorsque l'utilisateur exécute la requête, soit au niveau défini dans le groupe de travail.

Les niveaux disponibles sont les suivants :

- Basique – Chiffrement côté serveur Amazon S3 avec des clés gérées par Amazon S3 (SSE-S3).
- Intermédiaire – Chiffrement côté serveur avec des clés gérées par KMS (SSE_KMS)
- Avancé – Chiffrement côté client avec des clés gérées par KMS (CSE-KMS).

Considérations et restrictions

- La fonctionnalité de chiffrement minimal n'est pas disponible pour les groupes de travail compatibles avec Apache Spark.
- La fonctionnalité de chiffrement minimal ne fonctionne que lorsque le groupe de travail n'active pas l'option Remplacer les paramètres côté client.
- Si l'option Remplacer les paramètres côté client est activée dans le groupe de travail, le paramètre de chiffrement du groupe de travail prévaut et le paramètre de chiffrement minimal n'a aucun effet.
- L'activation de cette fonctionnalité est gratuite.

Activation du chiffrement minimal pour un groupe de travail

Vous pouvez activer un niveau de chiffrement minimal pour les résultats des requêtes provenant de votre groupe de travail Athena SQL lorsque vous créez ou mettez à jour le groupe de travail. Pour ce faire, vous pouvez utiliser la console Athena, l'API Athena ou. AWS CLI

Utilisation de la console Athena pour activer le chiffrement minimal

Pour commencer à créer ou à modifier votre groupe de travail à l'aide de la console Athena, consultez Créer un groupe de travail ou Modifier un groupe de travail. Lors de la configuration de votre groupe de travail, suivez les étapes ci-dessous pour activer le chiffrement minimal.

Pour configurer le niveau de chiffrement minimal pour les résultats des requêtes des groupes de travail

1. Dans la section Configurations supplémentaires, développez Paramètres.
2. Désactivez l'option Remplacer les paramètres côté client ou vérifiez qu'elle n'est pas sélectionnée.
3. Dans la section Configurations supplémentaires, développez Configuration des résultats des requêtes.
4. Sélectionnez l'option Chiffrer les résultats des requêtes.
5. Dans Type de chiffrement, sélectionnez la méthode de chiffrement que vous souhaitez qu'Athena utilise pour les résultats des requêtes de votre groupe de travail (SSE_S3, SSE_KMS ou CSE_KMS). Ces types de chiffrement correspondent aux niveaux de sécurité basique, intermédiaire et avancé.

6. Pour appliquer la méthode de chiffrement que vous avez choisie comme niveau de chiffrement minimal pour tous les utilisateurs, sélectionnez Définir ***`encryption_method`*** comme niveau de chiffrement minimal.

   Lorsque vous sélectionnez cette option, un tableau indique la hiérarchie de chiffrement et les niveaux de chiffrement autorisés aux utilisateurs lorsque le type de chiffrement que vous choisissez devient le minimum.

7. Après avoir créé votre groupe de travail ou mis à jour la configuration de votre groupe de travail, choisissez Créer un groupe de travail ou Enregistrer les modifications.

À l'aide de l'API Athena ou AWS CLI pour activer un chiffrement minimal

Lorsque vous utilisez l'UpdateWorkGroupAPI CreateWorkGroupor pour créer ou mettre à jour un groupe de travail Athena SQL, définissez EnforceWorkGroupConfigurationsur `falsetrue`, EnableMinimumEncryptionConfigurationet utilisez le EncryptionOptionpour spécifier le type de chiffrement.

Dans le AWS CLI, utilisez la `update-work-group`commande `create-work-group`ou avec les `--configuration-updates` paramètres `--configuration` or et spécifiez les options correspondant à celles de l'API.

Utilisation des groupes de travail Athena compatibles avec IAM Identity Center

La fonction de propagation fiable des identités AWS IAM Identity Center permet d'utiliser les identités de vos employés dans tous les services AWS d'analyse. La propagation d’identité approuvée vous évite d’avoir à effectuer des configurations de fournisseur d’identité spécifiques au service ou des configurations de rôles IAM.

Avec IAM Identity Center, vous pouvez gérer la sécurité de connexion pour les identités de vos employés, également appelées utilisateurs employés. IAM Identity Center fournit un endroit unique où vous pouvez créer ou connecter les utilisateurs du personnel et gérer de manière centralisée leur accès à tous leurs AWS comptes et applications. Vous pouvez utiliser des autorisations multi-comptes pour attribuer l’accès aux Comptes AWSà ces utilisateurs. Vous pouvez utiliser les affectations d’application pour attribuer à vos utilisateurs l’accès aux applications compatibles avec IAM Identity Center, aux applications cloud et aux applications client SAML 2.0 (Security Assertion Markup Language). Pour plus d’informations, consultez la rubrique Trusted identity propagation across applications dans le Guide de l’utilisateur AWS IAM Identity Center .

Actuellement, la prise en charge d'Athena SQL pour la propagation d'identité approuvée vous permet d'utiliser la même identité pour Amazon EMR Studio et l'interface Athena SQL dans EMR Studio. Pour utiliser les identités IAM Identity Center avec Athena SQL dans EMR Studio, vous devez créer des groupes de travail compatibles avec IAM Identity Center dans Athena. Vous pouvez alors utiliser la console ou l'API IAM Identity Center pour attribuer des utilisateurs ou des groupes IAM Identity Center aux groupes de travail Athena compatibles avec IAM Identity Center. Les requêtes provenant d'un groupe de travail Athena qui utilise la propagation d'identité approuvée doivent être exécutées à partir de l'interface SQL Athena dans un EMR Studio sur lequel IAM Identity Center est activé.

Considérations et restrictions

Lorsque vous utilisez la propagation d'identité approuvée avec Amazon Athena, tenez compte des points suivants :

- Vous ne pouvez pas modifier la méthode d'authentification du groupe de travail après sa création.
  - Vous ne pouvez pas modifier les groupes de travail Athena SQL existants pour qu'ils prennent en charge les groupes de travail compatibles avec IAM Identity Center.
  - Vous ne pouvez pas modifier les groupes de travail compatibles IAM Identity Center pour qu'ils prennent en charge les autorisations IAM au niveau des ressources ou les politiques IAM basées sur l'identité.
- Pour accéder aux groupes de travail compatibles avec la propagation d'identités fiables, les utilisateurs d'IAM Identity Center doivent être affectés à `IdentityCenterApplicationArn` ce qui est renvoyé par la réponse de l'action de l'API GetWorkGroupAthena.
- Les autorisations d'accès Amazon S3 doivent être configurées pour utiliser la propagation d'identité approuvée. Pour plus d'informations, consultez la rubrique S3 Access Grants and corporate directory identities dans le Guide de l'utilisateur Amazon S3.
- Les groupes de travail Athena compatibles avec IAM Identity Center nécessitent que Lake Formation soit configuré pour utiliser les identités IAM Identity Center. Pour obtenir des informations sur la configuration, consultez la rubrique Integrating IAM Identity Center dans le Guide du développeur AWS Lake Formation .
- Par défaut, les requêtes expirent au bout de 30 minutes dans les groupes de travail qui utilisent la propagation d'identité approuvée. Vous pouvez demander une augmentation du délai d'expiration des requêtes, mais le délai maximum des requêtes dans les groupes de travail utilisant la propagation d'identité approuvée est d'une heure.
- La prise en compte des modifications des droits des utilisateurs ou des groupes dans les groupes de travail utilisant la propagation d'identité approuvée peuvent prendre jusqu'à une heure.

- Les requêtes d'un groupe de travail Athena utilisant la propagation d'identité approuvée ne peuvent pas être exécutées directement depuis la console Athena. Elles doivent être exécutées depuis l'interface Athena dans un EMR Studio sur lequel IAM Identity Center est activé. Pour plus d'informations sur l'utilisation d'Athena dans EMR Studio, consultez la rubrique Use the Amazon Athena SQL editor in EMR Studio dans le Guide de gestion Amazon EMR.
- La propagation d'identité approuvée n'est pas compatible avec les fonctionnalités Athena suivantes.
  - Clés de contexte `aws:CalledVia`.
  - Groupes de travail Athena pour Spark.
  - Accès fédéré à l'API Athena.
  - Accès fédéré à Athena à l'aide de Lake Formation et des pilotes JDBC et ODBC d'Athena.
- Vous pouvez utiliser la propagation d'identité sécurisée avec Athena uniquement dans les cas suivants : Régions AWS
  - `us-east-2` – USA Est (Ohio)
  - `us-east-1` – USA Est (Virginie du Nord)
  - `us-west-1` – USA Ouest (Californie du Nord)
  - `us-west-2` – USA Ouest (Oregon)
  - `af-south-1` – Afrique (Le Cap)
  - `ap-east-1` – Asie-Pacifique (Hong Kong)
  - `ap-southeast-3` – Asie-Pacifique (Jakarta)
  - `ap-south-1` – Asie-Pacifique (Mumbai)
  - `ap-northeast-3` – Asie-Pacifique (Osaka)
  - `ap-northeast-2` – Asie-Pacifique (Séoul)
  - `ap-southeast-1` – Asie-Pacifique (Singapour)
  - `ap-southeast-2` – Asie-Pacifique (Sydney)
  - `ap-northeast-1` – Asie-Pacifique (Tokyo)
  - `ca-central-1` – Canada (Centre)
  - `eu-central-1` – Europe (Francfort)
  - `eu-west-1` – Europe (Irlande)
  - `eu-west-2` – Europe (Londres)
  - `eu-south-1` – Europe (Milan)

- `eu-west-3` – Europe (Paris)
- `eu-north-1` – Europe (Stockholm)
- `me-south-1` – Moyen-Orient (Bahreïn)
- `sa-east-1` – Amérique du Sud (São Paulo)

Autorisations nécessaires

Les politiques suivantes doivent être attachées à l'utilisateur IAM de l'administrateur qui crée le groupe de travail compatible avec IAM Identity Center dans la console Athena.

- La politique gérée `AmazonAthenaFullAccess`. Pour plus de détails, consultez AWS politique gérée : AmazonAthenaFullAccess.
- La politique en ligne suivante qui autorise les actions IAM et IAM Identity Center :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:createRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "iam:ListRoles",
                "iam:PassRole",
                "identitystore:ListUsers",
                "identitystore:ListGroups",
                "identitystore:CreateUser",
                "identitystore:CreateGroup",
                "sso:ListInstances",
                "sso:CreateInstance",
                "sso:DeleteInstance",
                "sso:DescribeUser",
                "sso:DescribeGroup",
                "sso:ListTrustedTokenIssuers",
                "sso:DescribeTrustedTokenIssuer",
                "sso:ListApplicationAssignments",
                "sso:DescribeRegisteredRegions",
                "sso:GetManagedApplicationInstance",
                "sso:GetSharedSsoConfiguration",
                "sso:PutApplicationAssignmentConfiguration",
```

```
                "sso:CreateApplication",
                "sso:DeleteApplication",
                "sso:PutApplicationGrant",
                "sso:PutApplicationAuthenticationMethod",
                "sso:PutApplicationAccessScope",
                "sso:ListDirectoryAssociations",
                "sso:CreateApplicationAssignment",
                "sso:DeleteApplicationAssignment",
                "organizations:ListDelegatedAdministrators",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization",
                "organizations:CreateOrganization",
                "sso-directory:SearchUsers",
                "sso-directory:SearchGroups",
                "sso-directory:CreateUser"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Création d'un groupe de travail Athena compatible avec IAM Identity Center

La procédure suivante décrit les étapes et les options associées à la création d'un groupe de travail Athena compatible avec IAM Identity Center. Pour obtenir une description des autres options de configuration disponibles pour les groupes de travail Athena, consultez Créer un groupe de travail.

Création d'un groupe de travail avec SSO activée dans la console Athena

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans le panneau de navigation de la console Athena, choisissez Workgroups (Groupes de travail).
3. Sur la page Workgroups (Groupes de travail), choisissez Create workgroup (Créer un groupe de travail).
4. Sur la page Créer un groupe de travail, pour Nom du groupe de travail, saisissez le nom du groupe de travail.
5. Pour Moteur d'analytique, utilisez Athena SQL par défaut.

6. Pour Authentification, choisissez IAM Identity Center.
7. Pour Fonction du service en ce qui concerne l'accès à IAM Identity Center, choisissez une fonction du service existante ou créez-en une.

   Athena a besoin d'autorisations pour accéder à IAM Identity Center en votre nom. Pour ce faire, Athena doit avoir une fonction du service. Un rôle de service est un rôle IAM que vous gérez et qui autorise un AWS service à accéder à d'autres AWS services en votre nom. Pour plus d'informations, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service dans le Guide de l'utilisateur IAM.
8. Développez Configuration des résultats de requêtes, puis saisissez ou choisissez un chemin Amazon S3 pour Emplacement du résultat de la requête.
9. (Facultatif) Choisissez Chiffrer les résultats de requête.
10. (Facultatif) Choisissez Créer un préfixe S3 basé sur l'identité utilisateur.

    Lorsque vous créez un groupe de travail compatible avec IAM Identity Center, l'option Activer les autorisations d'accès S3 est sélectionnée par défaut. Vous pouvez utiliser les autorisations d'accès Amazon S3 pour contrôler l'accès aux emplacements des résultats de requête Athena (préfixes) dans Amazon S3. Pour plus d'informations sur les autorisations d'accès Amazon S3, consultez Managing access with S3 Access Grants.

    Dans les groupes de travail Athena qui utilisent l'authentification IAM Identity Center, vous pouvez activer la création d'emplacements de résultats de requête basés sur l'identité et gouvernés par les autorisations d'accès Amazon S3. Ces préfixes Amazon S3 basés sur l'identité utilisateur permettent aux utilisateurs d'un groupe de travail Athena d'isoler les résultats de leurs requêtes des autres utilisateurs du même groupe de travail.

    Lorsque vous activez l'option de préfixe utilisateur, Athena ajoute l'ID utilisateur en tant que préfixe de chemin Amazon S3 à l'emplacement de sortie des résultats de requête pour le groupe de travail (par exemple, `s3://DOC-EXAMPLE-BUCKET/${user_id}`). Pour utiliser cette fonctionnalité, vous devez configurer les autorisations d'accès pour autoriser uniquement l'utilisateur à accéder à l'emplacement portant le préfixe `user_id`. Pour un exemple de politique de rôle de localisation Amazon S3 Access Grants qui restreint l'accès aux résultats des requêtes Athena, consultez. Exemple de politique de rôle

> **Note**
>
> Lorsque l'option de préfixe S3 de l'identité utilisateur est sélectionnée, l'option de remplacement des paramètres côté client est automatiquement activée pour le groupe de travail, comme décrit à l'étape suivante. L'option de remplacement des paramètres côté client est requise pour la fonctionnalité de préfixe de l'identité utilisateur.

11. Développez Paramètres, puis vérifiez que l'option Remplacer les paramètres côté client est sélectionnée.

    Lorsque vous sélectionnez l'option Remplacer les paramètres côté client, les paramètres du groupe de travail sont appliqués au niveau du groupe de travail pour tous les clients du groupe de travail. Pour plus d'informations, consultez Les paramètres du groupe de travail remplacent les paramètres côté client.
12. (Facultatif) Définissez tous les autres paramètres de configuration dont vous avez besoin, comme décrit dans Créer un groupe de travail.
13. Choisissez Créer un groupe de travail.
14. Utilisez la section Groupes de travail de la console Athena pour attribuer des utilisateurs ou des groupes de votre répertoire IAM Identity Center à votre groupe de travail Athena compatible avec IAM Identity Center.

Exemple de politique de rôle

L'exemple suivant montre une politique relative à l'attachement d'un rôle à un emplacement Amazon S3 Access Grant qui restreint l'accès aux résultats des requêtes Athena.

```
{
    "Statement": [{
        "Action": ["s3:*"],
        "Condition": {
            "ArnNotEquals": {
                "s3:AccessGrantsInstanceArn": "arn:aws:s3:${region}:${account}:access-
grants/default"
            },
            "StringNotEquals": {
                "aws:ResourceAccount": "${account}"
            }
        },
```

```
        "Effect": "Deny",
        "Resource": "*",
        "Sid": "ExplicitDenyS3"
    }, {
        "Action": ["kms:*"],
        "Effect": "Deny",
        "NotResource": "arn:aws:kms:${region}:${account}:key/${keyid}",
        "Sid": "ExplictDenyKMS"
    }, {
        "Action": ["s3:ListMultipartUploadParts", "s3:GetObject"],
        "Condition": {
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": "arn:aws:s3:${region}:${account}:access-
grants/default"
            },
            "StringEquals": {
                "aws:ResourceAccount": "${account}"
            }
        },
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::ATHENA-QUERY-RESULT-LOCATION/${identitystore:UserId}/
*",
        "Sid": "ObjectLevelReadPermissions"
    }, {
        "Action": ["s3:PutObject", "s3:AbortMultipartUpload"],
        "Condition": {
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": "arn:aws:s3:${region}:${account}:access-
grants/default"
            },
            "StringEquals": {
                "aws:ResourceAccount": "${account}"
            }
        },
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::ATHENA-QUERY-RESULT-LOCATION/${identitystore:UserId}/
*",
        "Sid": "ObjectLevelWritePermissions"
    }, {
        "Action": "s3:ListBucket",
        "Condition": {
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": "arn:aws:s3:${region}:${account}:access-
grants/default"
```

```
            },
            "StringEquals": {
                "aws:ResourceAccount": "${account}"
            },
            "StringLikeIfExists": {
                "s3:prefix": ["${identitystore:UserId}", "${identitystore:UserId}/*"]
            }
        },
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::ATHENA-QUERY-RESULT-LOCATION",
        "Sid": "BucketLevelReadPermissions"
    }, {
        "Action": ["kms:GenerateDataKey", "kms:Decrypt"],
        "Effect": "Allow",
        "Resource": "arn:aws:kms:${region}:${account}:key/${keyid}",
        "Sid": "KMSPermissions"
    }],
    "Version": "2012-10-17"
}
```

API de groupes de travail Athena

Voici quelques-unes des opérations d'API REST utilisées pour les groupes de travail Athena. Dans toutes les opérations suivantes à l'exception de `ListWorkGroups`, vous devez spécifier un groupe de travail. Dans d'autres opérations, par exemple `StartQueryExecution`, le paramètre du groupe de travail est facultatif et les opérations ne sont pas répertoriées ici. Pour la liste complète des opérations, consultez la Référence API Amazon Athena.

- CreateWorkGroup
- DeleteWorkGroup
- GetWorkGroup
- ListWorkGroups
- UpdateWorkGroup

Dépannage des groupes de travail

Utilisez les conseils suivants pour dépanner des groupes de travail.

- Vérifiez les autorisations pour chaque utilisateur dans votre compte. Ils doivent avoir accès à l'emplacement pour les résultats des requêtes et au groupe de travail dans lequel ils souhaitent

exécuter des requêtes. S'ils veulent changer de groupes de travail, ils ont également besoin d'autorisations pour les deux groupes de travail. Pour plus d'informations, veuillez consulter Politiques IAM pour l'accès aux groupes de travail.

- Faites attention au contexte dans la console Athena, pour voir dans quel groupe de travail vous allez exécuter les requêtes. Si vous utilisez le pilote, assurez-vous de définir le groupe de travail comme celui dont vous avez besoin. Pour plus d'informations, veuillez consulter the section called "Spécifier un groupe de travail dans lequel exécuter des requêtes".
- Si vous utilisez l'API ou les pilotes pour exécuter des requêtes, vous devez spécifier l'emplacement des résultats de la requête de l'une des manières suivantes : pour les requêtes individuelles, utilisez OutputLocation(côté client). Dans le groupe de travail, utilisez WorkGroupConfiguration. Si l'emplacement n'est pas spécifié de l'une ou l'autre manière, Athena génère une erreur au moment de l'exécution de la requête.
- Si vous remplacez les paramètres côté client avec les paramètres du groupe de travail, vous pouvez rencontrer des erreurs avec l'emplacement de vos résultats de requête. Par exemple, il est possible que l'utilisateur d'un groupe de travail n'ait pas l'autorisation d'accéder à l'emplacement du groupe de travail dans Simple Storage Service (Amazon S3) pour stocker les résultats des requêtes. Dans ce cas, ajoutez les autorisations nécessaires.
- Les groupes de travail présentent des changements dans le comportement des opérations d'API. Les appels aux opérations API existantes suivantes nécessitent que les utilisateurs de votre compte aient des autorisations basées sur les ressources dans IAM pour les groupes de travail dans lesquels ils effectuent ces appels. S'il n'existe aucune autorisation d'accès au groupe de travail et aux actions de groupe de travail, les actions d'API suivantes génèrent `AccessDeniedException` : CreateNamedQueryDeleteNamedQueryGetNamedQuery, ListNamedQueries, StartQueryExecution,, StopQueryExecution, ListQueryExecutionsGetQueryExecutionGetQueryResults, et GetQueryResultsStream(cette action d'API n'est disponible que pour une utilisation avec le pilote et n'est pas exposée autrement à un usage public). Pour plus d'informations, consultez la rubrique Actions, ressources et clés de condition pour Amazon Athena dans la section Référence de l'autorisation de service.

  Les appels aux opérations d'BatchGetNamedQueryAPI BatchGetQueryExecutionet d'API renvoient des informations uniquement sur les requêtes exécutées dans des groupes de travail auxquels les utilisateurs ont accès. Si l'utilisateur n'a pas accès au groupe de travail, ces opérations d'API renvoient les ID de requête non autorisés dans le cadre de la liste des ID non traités. Pour plus d'informations, consultez the section called " API de groupes de travail Athena".
- Si le groupe de travail dans lequel une requête sera exécutée est configuré avec un emplacement imposé pour les résultats de la requête, ne spécifiez pas de `external_location` pour la requête

CTAS. Athena émet une erreur et fait échouer une requête qui spécifie un `external_location` dans ce cas. Par exemple, cette requête échoue si vous remplacez les paramètres côté client par l'emplacement de vos résultats de requête, ce qui force le groupe de travail à utiliser son propre emplacement : `CREATE TABLE <DB>.<TABLE1> WITH (format='Parquet', external_location='s3://DOC-EXAMPLE-BUCKET/test/') AS SELECT * FROM <DB>.<TABLE2> LIMIT 10;`

Les erreurs suivantes peuvent s'afficher. Ce tableau fournit une liste de certaines des erreurs liées aux groupes de travail et suggère des solutions.

Erreurs de groupe de travail

Erreur	Se produit lorsque...
query state CANCELED (état de la requête ANNULÉ). Bytes scanned limit was exceeded (La limite d'octets analysés a été dépassée).	Une requête atteint une limite de données par requête et est annulée. Envisagez de réécrire la requête afin qu'elle lise moins de données, ou contactez votre administrateur de compte.
L'utilisateur : arn:aws:iam : :123456789012:user/abc n'est pas autorisé à exécuter : athena : on resource : arn:aws:athena:us-east- 1:123456789012:workgroup/workgroupname StartQueryExecution	Un utilisateur exécute une requête dans un groupe de travail, mais n'y a pas accès. Mettez à jour votre politique pour avoir accès au groupe de travail.
SAISIE_INVALIDE. WorkGroup <name>est désactivé.	Un utilisateur exécute une requête dans un groupe de travail, mais le groupe de travail est désactivé. Votre groupe de travail peut être désactivé par votre administrateur. Il est également possible que vous n'y ayez pas accès. Dans les deux cas, contactez un administrateur y ayant accès pour modifier les groupes de travail.
SAISIE_INVALIDE. WorkGroup <name>n'est pas trouvé.	Un utilisateur exécute une requête dans un groupe de travail, mais le groupe de travail

Erreur	Se produit lorsque...
	n'existe pas. Cela peut se produire si le groupe de travail a été supprimé. Basculer vers un autre groupe de travail pour exécuter votre requête.
InvalidRequestException: lors de l'appel de l' StartQueryExecutionopération : aucun emplacement de sortie n'est fourni. An output location is required either through the Workgroup result configuration setting or as an API input (Un emplacement de sortie est nécessaire, soit par le biais du paramètre de configuration des résultats du groupe de travail, soit en tant qu'entrée API).	Un utilisateur exécute une requête avec l'API sans spécifier l'emplacement pour les résultats de la requête. Vous devez définir l'emplace ment de sortie pour les résultats des requêtes de l'une des deux manières suivantes : soit pour les requêtes individuelles, en utilisant OutputLocation(côté client), soit dans le groupe de travail, en utilisant. WorkGroupConfiguration
The Create Table As Select query failed because it was submitted with an 'external _location' property to an Athena Workgroup that enforces a centralized output location for all queries (La requête « Create Table As Select » a échoué parce qu'elle a été soumise avec une propriété «◦external_location◦» à un groupe de travail Athena qui impose un emplacement de sortie centralisé pour toutes les requêtes). Please remove the 'external_location' property and resubmit the query (Supprimez la propriété «◦external_location◦» et soumettez à nouveau la requête).	Si le groupe de travail dans lequel une requête s'exécute est configuré avec un emplacement imposé pour les résultats de la requête, et que vous spécifiez un `external_location` pour la requête CTAS. Dans ce cas, supprimez le `external_location` et exécutez à nouveau la requête.

Erreur	Se produit lorsque...
Cannot create prepared statement *prepared_statement_name* (Impossible de créer l'instruction préparée « prepared_statement _name »). The number of prepared statement s in this workgroup exceeds the limit of 1000 (Le nombre d'instructions préparées dans ce groupe de travail dépasse la limite de 1◦000).	Le groupe de travail contient plus que la limite de 1 000 instructions préparées. Pour résoudre ce problème, utilisez DEALLOCATE PREPARE pour supprimer une ou plusieurs instructions préparées du groupe de travail. Vous pouvez également créer un nouveau groupe de travail.

Contrôle des coûts et surveillance des requêtes à l'aide de CloudWatch métriques et d'événements

Les groupes de travail vous permettent de définir des limites de contrôle de l'utilisation des données par requête ou par groupe de travail, de configurer des alarmes lorsque ces limites sont dépassées et de publier des métriques de requête sur. CloudWatch

Dans chaque groupe de travail, vous pouvez :

- Configurer des Data usage controls (Contrôles d'utilisation des données) par requête et par groupe de travail, et mettre en place des actions si les requêtes dépassent les seuils.
- Affichez et analysez les métriques des requêtes, puis publiez-les sur CloudWatch. Si vous créez un groupe de travail dans la console, le paramètre de publication des métriques CloudWatch est sélectionné pour vous. Si vous utilisez les opérations d'API, vous devez activer la publication des métriques. Une fois les métriques publiées, elles s'affichent dans l'onglet Metrics (Métriques) du panneau Workgroups (Groupes de travail). Les métriques sont désactivées par défaut pour le groupe de travail principal.

Vidéo

La vidéo suivante montre comment créer des tableaux de bord personnalisés et définir des alarmes et des déclencheurs sur les métriques dans CloudWatch. Vous pouvez utiliser des tableaux de bord préremplis directement à partir de la console Athena pour utiliser ces métriques de requête.

Surveillance des requêtes Amazon Athena à l'aide d'Amazon CloudWatch

Rubriques

- Activation des métriques de CloudWatch requête
- Surveillance des requêtes Athena à l'aide de métriques CloudWatch
- Surveillance des requêtes Athena à l'aide des événements Amazon EventBridge
- Surveillance des métriques d'utilisation d'Athena
- Définition des limites pour le contrôle d'utilisation des données

Activation des métriques de CloudWatch requête

Lorsque vous créez un groupe de travail dans la console, le paramètre de publication des métriques de requête vers CloudWatch est sélectionné par défaut.

Activation ou désactivation des métriques de requête dans la console Athena pour un groupe de travail

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le panneau de navigation, choisissez Workgroups (Groupes de travail).
4. Choisissez le lien du groupe de travail à modifier.
5. Sur la page des détails du groupe de travail, choisissez Edit (Modifier).
6. Dans la section Paramètres, sélectionnez ou désactivez Publier les métriques de requête sur AWS CloudWatch.

Si vous utilisez les opérations d'API, l'interface de ligne de commande ou l'application client avec le pilote JDBC pour créer des groupes de travail, afin de permettre la publication des métriques de requête, définissez `PublishCloudWatchMetricsEnabled` ce paramètre sur in. `true` WorkGroupConfiguration L'exemple suivant illustre uniquement la configuration des métriques et fait abstraction de toute autre configuration :

```
"WorkGroupConfiguration": {
      "PublishCloudWatchMetricsEnabled": "true"
     ....
     }
```

Surveillance des requêtes Athena à l'aide de métriques CloudWatch

Athena publie les métriques relatives aux requêtes sur Amazon CloudWatch, lorsque l'option Publier les métriques de requête sur est sélectionnée. CloudWatch Vous pouvez créer des tableaux de bord personnalisés, définir des alarmes et des déclencheurs sur les métriques ou utiliser des tableaux de bord préremplis directement depuis la console Athena. CloudWatch

Lorsque vous activez des métriques de requête pour des requêtes dans les groupes de travail, les métriques sont affichées dans l'onglet Metrics (Métriques) du panneau Workgroups (Groupes de travail) pour chaque groupe de travail de la console Athena.

Athena publie les métriques suivantes sur la CloudWatch console :

- `DPUAllocated` : le nombre total de DPU (unités de traitement des données) allouées à une réserve de capacité pour exécuter des requêtes.
- `DPUConsumed` : le nombre de DPU activement consommées par les requêtes étant dans l'état `RUNNING` à un moment donné dans une réserve. Métrique émise uniquement lorsque le groupe de travail est associé à une réserve de capacité et inclut tous les groupes de travail associés à une réserve.
- `DPUCount` : le nombre maximum de DPU consommées par votre requête, publié une seule fois à la fin de la requête.
- `EngineExecutionTime` : le nombre de millisecondes nécessaires à l'exécution de la requête.
- `ProcessedBytes` : le nombre d'octets qu'Athena a analysé par requête DML.
- `QueryPlanningTime` : le nombre de millisecondes nécessaires à Athena pour planifier le flux de traitement des requêtes.
- `QueryQueueTime` : le nombre de millisecondes pendant lesquelles la requête est restée dans la file d'attente des ressources.

- `ServicePreProcessingTime` : le nombre de millisecondes nécessaires à Athena pour prétraiter la requête avant de la soumettre au moteur de requête.
- `ServiceProcessingTime` : le nombre de millisecondes nécessaires à Athena pour traiter les résultats de la requête après que le moteur de requête ait fini d'exécuter la requête.
- `TotalExecutionTime` : le nombre de millisecondes nécessaires à Athena pour exécuter une requête DDL ou DML.

Pour des descriptions plus complètes, veuillez consulter les rubriques Liste des CloudWatch métriques et des dimensions d'Athena plus avant dans le présent document.

Ces métriques ont les dimensions suivantes :

- `CapacityReservation` : le nom de la réserve de capacité utilisée pour exécuter la requête, le cas échéant.
- `QueryState` – `SUCCEEDED`, `FAILED`, ou `CANCELED`
- `QueryType` – `DML`, `DDL`, ou `UTILITY`
- `WorkGroup` – nom du groupe de travail

Athena publie la métrique suivante sur la CloudWatch console sous l'espace de `AmazonAthenaForApacheSpark` noms :

- `DPUCount` – nombre de DPU consommés au cours de la session pour exécuter les calculs.

Cette métrique a les dimensions suivantes :

- `SessionId` – L'ID de la session dans laquelle les calculs sont soumis.
- `WorkGroup` – nom du groupe de travail.

Pour de plus amples informations, veuillez consulter Liste des CloudWatch métriques et des dimensions d'Athena plus loin dans cette rubrique. Pour plus d'informations sur les métriques d'utilisation d'Athena, veuillez consulter Surveillance des métriques d'utilisation d'Athena.

Pour afficher les métriques de requête d'un groupe de travail dans la console.

1. Ouvrez la console Athena à l’adresse https://console.aws.amazon.com/athena/.

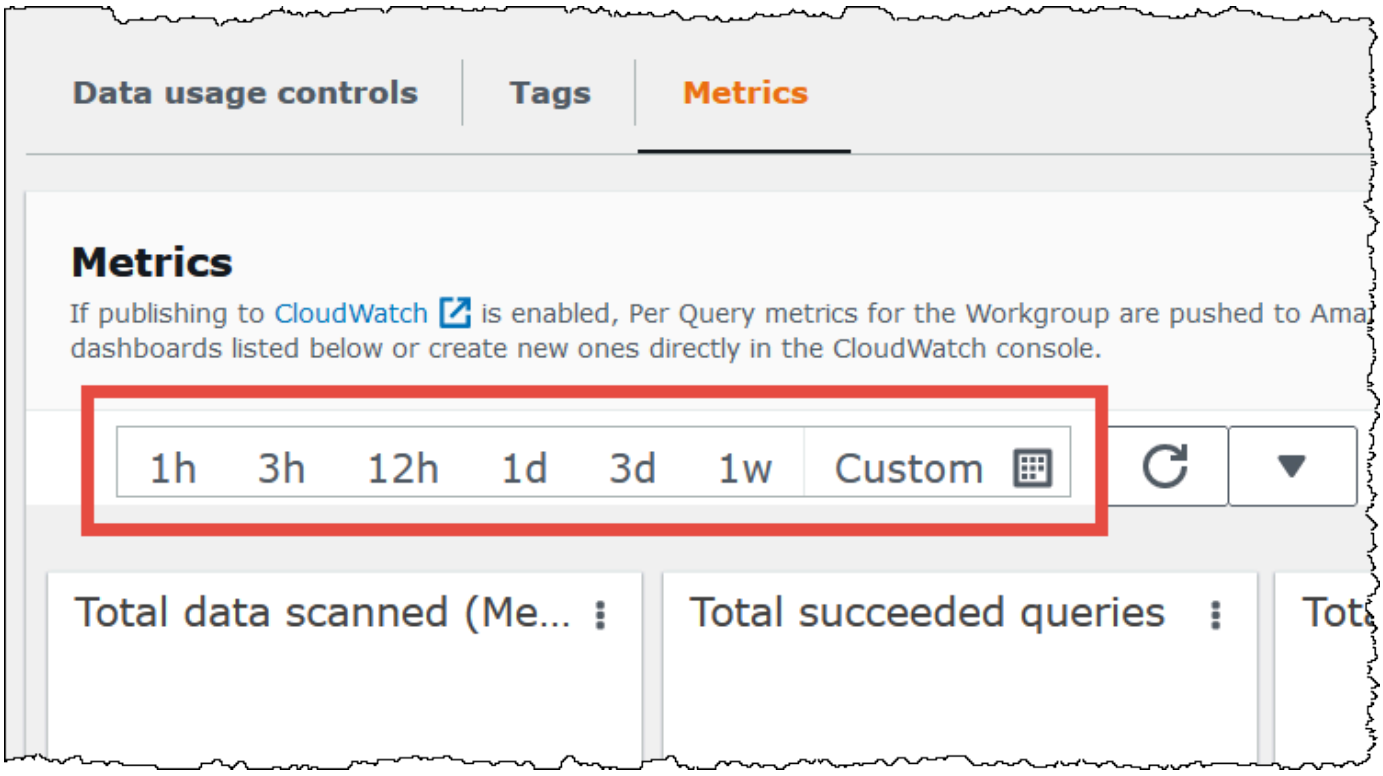
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le panneau de navigation, choisissez Workgroups (Groupes de travail).
4. Choisissez le groupe de travail souhaité dans la liste, puis choisissez l'onglet Metrics (Métriques).

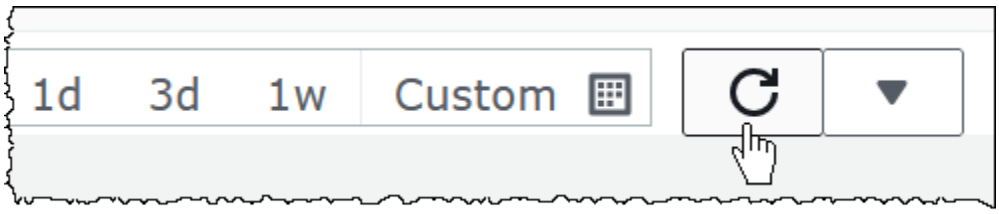
   Le tableau de bord des métriques s'affiche.

   > **Note**
   >
   > Si vous venez d'activer les métriques pour le groupe de travail et/ou s'il n'y a pas eu d'activité de requête récente, les graphiques du tableau de bord peuvent être vides. L'activité de requête est extraite CloudWatch en fonction de l'intervalle que vous spécifiez à l'étape suivante.

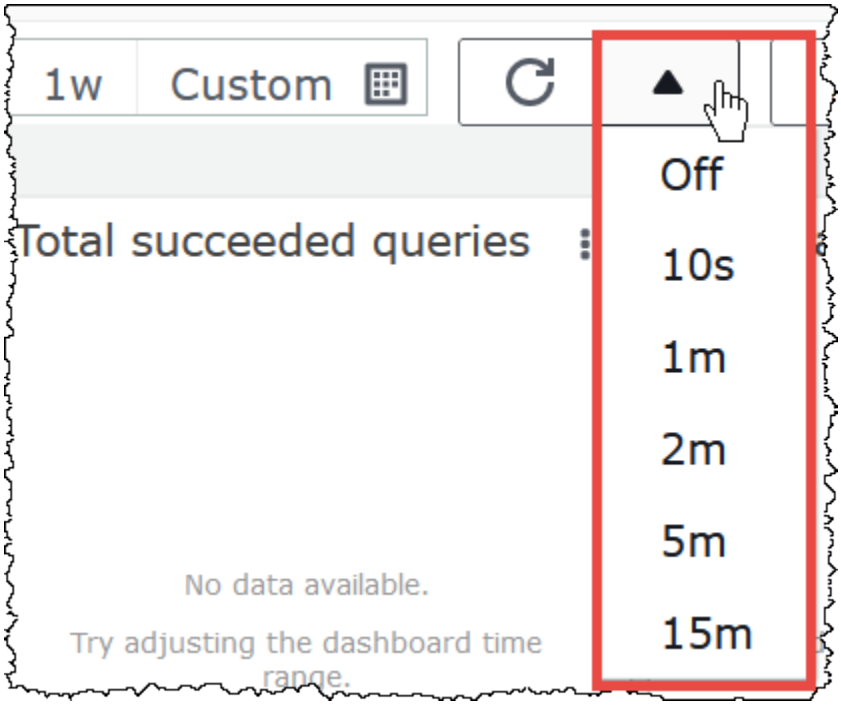
5. Dans la section Metrics, choisissez l'intervalle de métriques qu'Athena doit utiliser pour récupérer les métriques de requête CloudWatch, ou spécifiez un intervalle personnalisé.



6. Pour actualiser les métriques affichées, choisissez l'icône Actualiser.



7. Cliquez sur la flèche à côté de l'icône d'actualisation pour choisir la fréquence à laquelle vous souhaitez que l'affichage des métriques soit mis à jour.



Pour consulter les statistiques dans la CloudWatch console Amazon

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation, sélectionnez Métriques, Toutes les métriques.
3. Sélectionnez l'espace de nom AWS/Athena.

Pour afficher les métriques grâce à la CLI

- Effectuez l’une des actions suivantes :
  - Pour répertorier les métriques d'Athena, ouvrez une invite de commandes et utilisez la commande suivante :

```
aws cloudwatch list-metrics --namespace "AWS/Athena"
```

  - Pour répertorier toutes les métriques disponibles, utilisez la commande suivante :

```
aws cloudwatch list-metrics"
```

Liste des CloudWatch métriques et des dimensions d'Athena

Si vous avez activé CloudWatch les métriques dans Athena, celle-ci envoie les métriques suivantes à CloudWatch chaque groupe de travail. Les métriques suivantes utilisent l'espace de noms `AWS/Athena`.

Nom des métriques	Description
DPUAllocated	Le nombre total de DPU (unités de traitement des données) allouées à une réserve de capacité pour exécuter des requêtes.
DPUConsumed	Le nombre de DPU activement consommées par les requêtes étant dans l'état `RUNNING` à un moment donné dans une réserve. Cette métrique n'est émise que lorsque le groupe de travail est associé à une réserve de capacité et inclut tous les groupes de travail associés à une réserve. Par conséquent, si vous déplacez un groupe de travail d'une réserve à une autre, la métrique inclut les données de la période pendant laquelle le groupe de travail appartenait à la première réserve. Pour plus d'informations sur les réserves de capacité, veuillez consulter Gestion de la capacité de traitement des requêtes.
DPUCount	Le nombre maximal de DPU consommées par votre requête, publié une seule fois à la fin de la requête. Cette métrique n'est émise que pour les groupes de travail associés à une réserve de capacité.
EngineExecutionTime	Le nombre de millisecondes nécessaires à l'exécution de la requête.
ProcessedBytes	Le nombre d'octets qu'Athena a analysé par requête DML. Pour les requêtes qui ont été annulées (soit par les utilisateurs, soit automatiquement, si la limite a été atteinte), cela inclut la quantité de données analysées avant l'heure de l'annulation. Cette métrique n'est pas signalée pour les requêtes DDL.
QueryPlanningTime	Le nombre de millisecondes nécessaires à Athena pour planifier le flux de traitement des requêtes. Cela inclut le temps passé à récupérer les partitions de la table à partir de la source de

Nom des métriques	Description
	données, Notez que, dans la mesure où le moteur de requêtes effectue la planification des requêtes, le temps de planification des requêtes est un sous-ensemble de EngineExecutionTime.
QueryQueueTime	Le nombre de millisecondes pendant lesquelles la requête est restée dans la file d'attente des ressources. Notez que si des erreurs transitoires se produisent, la requête peut être automatiq uement replacée dans la file d'attente.
ServicePreProcessingTime	Le nombre de millisecondes nécessaires à Athena pour prétraiter la requête avant de la soumettre au moteur de requête.
ServiceProcessingTime	Le nombre de millisecondes nécessaires à Athena pour traiter les résultats de la requête après que le moteur de requête ait fini d'exécuter la requête.
TotalExecutionTime	Le nombre de millisecondes nécessaires pour qu'Athena exécute une requête DDL ou DML. TotalExecutionTime inclut QueryQueu eTime QueryPlanningTime, EngineExecutionTime, et ServicePr ocessingTime.

Ces métriques pour Athena ont les dimensions suivantes.

Dimension	Description
CapacityReservation	Le nom de la réserve de capacité qui a été utilisée pour exécuter la requête, le cas échéant. Lorsqu'aucune réserve de capacité n'est utilisée, cette dimension ne renvoie aucune donnée.
QueryState	L'état de la requête. Statistiques valides : SUCCEEDED (réussite), FAILED (échec) ou CANCELED (annulé).
QueryType	Le type de requête.

Dimension	Description
	Statistiques valides : `DDL`, `DML` ou `UTILITY`. Type d'instruction de requête exécutée. `DDL` indique les instructions de requête DDL (Data Definition Language). `DML` indique les instructions de requête DML (Data Manipulation Language), telles que `CREATE TABLE AS SELECT`. `UTILITY` indique des instructions de requête autres que DDL et DML, telles que `SHOW CREATE TABLE` ou `DESCRIBE TABLE`.
WorkGroup	Le nom du groupe de travail.

Surveillance des requêtes Athena à l'aide des événements Amazon EventBridge

Vous pouvez utiliser Amazon Athena avec Amazon EventBridge pour recevoir des notifications en temps réel concernant l'état de vos requêtes. Lorsqu'une requête à laquelle vous avez soumis des états de transition, Athena publie un événement EventBridge contenant des informations sur cette transition d'état de requête. Vous pouvez écrire des règles simples pour les événements qui vous intéressent et effectuer des actions automatisées lorsqu'un événement correspond à une règle. Par exemple, vous pouvez créer une règle qui invoque une AWS Lambda fonction lorsqu'une requête atteint un état terminal. Les événements sont générés dans la mesure du possible.

Avant de créer des règles d'événement pour Athena, vous devez procéder comme suit :

- Familiarisez-vous avec les événements, les règles et les cibles dans EventBridge. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge ? Pour plus d'informations sur la configuration des règles, consultez Getting started with Amazon EventBridge.
- Créez la ou les cible(s) à utiliser dans vos règles d'événement.

> **Note**
>
> Athena propose actuellement un type d'événement appelé événement de changement d'état de requête Athena, mais peut ajouter d'autres types et détails d'événement. Si vous désérialisez par programmation les données JSON d'événement, veillez à ce que votre application soit prête à traiter des propriétés inconnues si ces propriétés supplémentaires sont ajoutées.

Format des événements Athena

Voici le modèle de base d'un événement Amazon Athena.

```
{
    "source":[
        "aws.athena"
    ],
    "detail-type":[
        "Athena Query State Change"
    ],
    "detail":{
        "currentState":[
            "SUCCEEDED"
        ]
    }
}
```

Événement de changement d'état de requête Athena

L'exemple suivant montre un événement de changement d'état de requête Athena dont la valeur `currentState` est SUCCEEDED.

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Athena Query State Change",
    "source":"aws.athena",
    "account":"123456789012",
    "time":"2019-10-06T09:30:10Z",
    "region":"us-east-1",
    "resources":[

    ],
    "detail":{
        "versionId":"0",
        "currentState":"SUCCEEDED",
        "previousState":"RUNNING",
        "statementType":"DDL",
        "queryExecutionId":"01234567-0123-0123-0123-012345678901",
        "workgroupName":"primary",
        "sequenceNumber":"3"
    }
```

```
}
```

L'exemple suivant montre un événement de changement d'état de requête Athena dont la valeur `currentState` est `FAILED`. Le bloc `athenaError` apparaît uniquement lorsque `currentState` est `FAILED`. Pour plus d'informations sur les valeurs de `errorCategory` et `errorType`, voir Catalogue d'erreurs Athena.

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Athena Query State Change",
    "source":"aws.athena",
    "account":"123456789012",
    "time":"2019-10-06T09:30:10Z",
    "region":"us-east-1",
    "resources":[
    ],
    "detail":{
        "athenaError": {
            "errorCategory": 2.0, //Value depends on nature of exception
            "errorType": 1306.0, //Type depends on nature of exception
            "errorMessage": "Amazon S3 bucket not found", //Message depends on nature
 of exception
            "retryable":false //Retryable value depends on nature of exception
        },
        "versionId":"0",
        "currentState": "FAILED",
        "previousState": "RUNNING",
        "statementType":"DML",
        "queryExecutionId":"01234567-0123-0123-0123-012345678901",
        "workgroupName":"primary",
        "sequenceNumber":"3"
    }
}
```

Propriétés de sortie

La sortie JSON inclut les propriétés suivantes.

Propriété	Description
`athenaError`	Apparaît uniquement lorsque `currentState` est `FAILED`. Contient des informations sur l'erreur qui s'est produite, notamment la catégorie d'erreur, le type d'erreur, le message d'erreur et la possibilité de réessayer l'action qui a conduit à l'erreur. Les valeurs de chacun de ces champs dépendent de la nature de l'erreur. Pour plus d'informations sur les valeurs de `errorCategory` et `errorType`, voir Catalogue d'erreurs Athena.
`versionId`	Numéro de version du schéma de l'objet détaillé.
`currentState`	État dans lequel la requête a été placée au moment de l'événement.
`previousState`	État initial de la requête au moment de l'événement.
`statementType`	Type d'instruction de requête exécutée.
`queryExecutionId`	Identifiant unique de la requête exécutée.
`workgroupName`	Nom du groupe de travail dans lequel la requête a été exécutée.
`sequenceNumber`	Nombre croissant de façon monotone qui permet de dédupliquer et d'ordonner les événements entrants qui impliquent une seule requête exécutée. Lorsque des événements dupliqués sont publiés pour le même changement d'état, la valeur `sequenceNumber` est la même. Lorsqu'une requête subit plusieurs changements d'état, par exemple des requêtes faisant l'objet d'un rare remplacement en file d'attente, vous pouvez utiliser `sequenceNumber` pour ordonner des événements avec des valeurs `currentState` et `previousState` identiques.

Exemple

L'exemple suivant publie des événements dans une rubrique Amazon SNS à laquelle vous êtes abonné. Lorsque Athena est interrogé, vous recevez un e-mail. L'exemple suppose que la rubrique Amazon SNS existe et que vous y êtes abonné.

Publication des événements Athena dans une rubrique Amazon SNS

1. Créez la cible pour votre rubrique Amazon SNS. `events.amazonaws.com`Autorisez le EventBridge responsable du service des événements à publier sur votre rubrique Amazon SNS, comme dans l'exemple suivant.

```
{
    "Effect":"Allow",
    "Principal":{
        "Service":"events.amazonaws.com"
    },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-1:111111111111:your-sns-topic"
}
```

2. Utilisez la AWS CLI `events put-rule` commande pour créer une règle pour les événements Athena, comme dans l'exemple suivant.

```
aws events put-rule --name {ruleName} --event-pattern '{"source": ["aws.athena"]}'
```

3. Utilisez la AWS CLI `events put-targets` commande pour associer la cible de la rubrique Amazon SNS à la règle, comme dans l'exemple suivant.

```
aws events put-targets --rule {ruleName} --targets Id=1,Arn=arn:aws:sns:us-
east-1:111111111111:your-sns-topic
```

4. Interrogez Athena et observez la cible invoquée. Vous devriez recevoir les e-mails correspondants à partir de la rubrique Amazon SNS.

Utilisation Notifications des utilisateurs AWS avec Amazon Athena

Vous pouvez utiliser Notifications des utilisateurs AWS pour configurer des canaux de diffusion afin d'être averti des événements Amazon Athena. Vous recevez une notification lorsqu'un événement correspond à une règle que vous avez spécifiée. Vous pouvez recevoir des notifications relatives à des événements via plusieurs canaux, notamment des e-mails, des notifications de chat AWS Chatbot ou des notifications push AWS Console Mobile Application. Vous pouvez également consulter les notifications dans le centre de notifications de la console. Notifications des utilisateurs prend en charge l'agrégation, ce qui peut réduire le nombre de notifications que vous recevez lors d'événements spécifiques.

Pour plus d’informations, consultez le Guide de l’utilisateur Notifications des utilisateurs AWS .

Surveillance des métriques d'utilisation d'Athena

Vous pouvez utiliser les statistiques CloudWatch d'utilisation pour avoir une idée de la manière dont votre compte utilise les ressources en affichant votre utilisation actuelle des services sur CloudWatch des graphiques et des tableaux de bord.

Pour Athena, les mesures de disponibilité d'utilisation correspondent aux Service AWS quotas pour Athena. Vous pouvez configurer des alarmes qui vous alertent lorsque votre utilisation approche d’un quota de service. Pour de plus amples informations sur les quotas de service Athena, consultez Service Quotas. Pour plus d'informations sur les statistiques AWS d'utilisation, consultez les statistiques AWS d'utilisation dans le guide de CloudWatch l'utilisateur Amazon.

Athena publie les métriques suivantes dans l'espace de noms `AWS/Usage`.

Nom des métriques	Description
`ResourceCount`	Somme de toutes les requêtes en file d'attente et en cours d'exécution Région AWS par compte, séparées par type de requête (DML ou DDL). Le maximum est la seule statistique utile pour cette métrique. Cette métrique est publiée périodiquement toutes les minutes. Si vous n'exécutez aucune requête, la métrique ne signale rien (même pas 0). La métrique est publiée uniquement si des requêtes actives sont en cours d'exécution au moment de la prise de la mesure.

Les dimensions suivantes permettent d'affiner les métriques d'utilisation publiées par Athena.

Dimension	Description
`Service`	Le nom du Service AWS conteneur de la ressource. Pour Athena, la valeur de cette dimension est au format `Athena`.

Dimension	Description
`Resource`	Type de ressource en cours d'exécution. La valeur de ressource pour l'utilisation de la requête Athena est `ActiveQueryCount` .
`Type`	Type d'entité faisant l'objet d'un rapport. Actuellement, la seule valeur valide pour les métriques d'utilisation d'Athena est `Resource`.
`Class`	Classe de ressource suivie. Pour Athena, `Class` peut être `DML` ou `DDL`.

Afficher les statistiques d'utilisation des ressources Athena dans la console CloudWatch

Vous pouvez utiliser la CloudWatch console pour consulter un graphique des indicateurs d'utilisation d'Athena et configurer des alarmes qui vous alertent lorsque votre utilisation approche un quota de service.

Pour afficher les mesures d'utilisation des ressources Athena

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation, sélectionnez Métriques, Toutes les métriques.
3. Choisissez Usage (Utilisation), puis By AWS Ressource (Par ressource AWS).

   La liste des métriques d'utilisation des Service Quotas s'affiche.
4. Cochez la case située à côté d'Athéna et. ActiveQueryCount
5. Sélectionnez l'onglet Graphed metrics (Graphiques des métriques).

   Le graphique ci-dessus montre votre utilisation actuelle de la AWS ressource.

Pour plus d'informations sur l'ajout de quotas de service au graphique et sur le paramétrage d'une alarme qui vous avertira si vous approchez du quota de service, consultez Visualisation de vos quotas de service et définition d'alarmes dans le guide de CloudWatch l'utilisateur Amazon. Pour plus d'informations sur la définition de limites d'utilisation par groupe de travail, consultez Définition des limites pour le contrôle d'utilisation des données.

Définition des limites pour le contrôle d'utilisation des données

Athena vous permet de définir deux types de contrôles des coûts : la limite par requête et la limite par groupe de travail. Pour chaque groupe de travail, vous pouvez définir une seule limite par requête et plusieurs limites par groupe de travail.

- La limite pour le contrôle par requête spécifie le volume total de données analysées par requête. Si une requête s'exécutant dans le groupe de travail dépasse la limite, elle est annulée. Vous pouvez créer une seule limite pour le contrôle par requête dans un groupe de travail qui s'applique à chaque requête exécutée dans ce dernier. Modifiez la limite si nécessaire. Pour obtenir des informations plus détaillées, consultez Pour créer un contrôle d'utilisation des données par requête.
- La limite pour le contrôle d'utilisation des données à l'échelle du groupe de travail spécifie le volume total de données analysées pour toutes les requêtes s'exécutant dans ce groupe de travail au cours de la période spécifiée. Vous pouvez créer plusieurs limites par groupe de travail. La limite de requêtes à l'échelle du groupe de travail vous permet de définir plusieurs seuils sur des volumes horaires ou quotidiens de données analysées par des requêtes s'exécutant dans le groupe de travail.

  Si la quantité globale de données analysées dépasse le seuil, vous pouvez envoyer une notification à une rubrique Amazon SNS. Configurez une alarme Amazon SNS et une action dans la console Athena pour informer un administrateur en cas d'utilisation hors limites. Pour obtenir des informations plus détaillées, consultez Pour créer un contrôle d'utilisation des données par groupe de travail. Vous pouvez également créer une alarme et une action sur n'importe quelle métrique publiée par Athena depuis la CloudWatch console. Par exemple, vous pouvez définir une alerte sur plusieurs requêtes en échec. Cette alerte peut déclencher l'envoi d'un e-mail à un administrateur si le nombre dépasse un certain seuil. En cas de dépassement de la limite, une action envoie une notification d'alarme Amazon SNS aux utilisateurs spécifiés.

  Autres actions que vous pouvez effectuer :

  - Invoque une fonction Lambda. Pour plus d'informations, consultez Invocation des fonctions Lambda en utilisant des notifications Amazon SNS dans le Guide du développeur Amazon Simple Notification Service.
  - Désactivation du groupe de travail pour arrêter l'exécution d'autres requêtes. Pour les étapes, consultez Activer et désactiver un groupe de travail.

Les limites par requête et par groupe de travail sont indépendantes les unes des autres. Une action spécifiée est exécutée à chaque dépassement de l'une des limites. Si deux ou plusieurs utilisateurs

exécutent des requêtes au même moment dans le même groupe de travail, il est possible que chacune des requêtes ne dépasse pas les limites spécifiées, mais que le volume total de données analysées dépasse la limite d'utilisation des données par groupe de travail. Dans ce cas, une alarme Amazon SNS est envoyée à l'utilisateur.

Pour créer un contrôle d'utilisation des données par requête

La limite pour le contrôle par requête spécifie le volume total de données analysées par requête. Si une requête s'exécutant dans le groupe de travail dépasse la limite, elle est annulée. Les requêtes annulées sont facturées conformément à la Tarification Amazon Athena.

> **Note**
>
> Dans le cas de requêtes annulées ou échouées, il est possible qu'Athena ait déjà écrit des résultats partiels sur Simple Storage Service (Amazon S3). Dans ce cas, Athena ne supprime pas les résultats partiels du préfixe Simple Storage Service (Amazon S3) où sont stockés les résultats. Vous devez supprimer le préfixe Simple Storage Service (Amazon S3) avec des résultats partiels. Athena utilise les téléchargements partitionnés Simple Storage Service (Amazon S3) pour écrire des données Simple Storage Service (Amazon S3). Nous vous recommandons de définir la politique du cycle de vie du compartiment pour interrompre les chargements partitionnés en cas d'échec des requêtes. Pour de plus amples informations, consultez la section Utilisation d'une politique de cycle de vie des compartiments pour l'interruption des chargements partitionnés inachevés dans le Guide de l'utilisateur Amazon Simple Storage Service.

Vous pouvez créer une seule limite pour le contrôle par requête dans un groupe de travail qui s'applique à chaque requête exécutée dans ce dernier. Modifiez la limite si nécessaire.

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le panneau de navigation, choisissez Groupes de travail.
4. Choisissez le nom du groupe de travail dans la liste.
5. Sur l'onglet Data usage controls (Contrôle d'utilisation des données), dans la section Per query data usage control (Contrôle de l'utilisation des données par requête), choisissez Manage (Gérer).
6. Sur la page Manage per query data usage control (Gestion du contrôle de l'utilisation des données par requête), spécifiez les valeurs suivantes :

   - Pour Data limit (Limite des données), spécifiez une valeur comprise entre 10 Mo (minimum) et 7 Eo (maximum).

   > **Note**
   >
   > Il s'agit des limites imposées par la console pour les contrôles d'utilisation des données dans les groupes de travail. Elles ne représentent pas les limites de requêtes dans Athena.

   - Pour les unités, sélectionnez la valeur de l'unité dans la liste déroulante (par exemple, Kilobytes KB (Kilooctets Ko) ou Exabytes EB (Exaoctets Eo).

   L'action par défaut consiste à annuler la requête si elle dépasse la limite. Ce paramètre ne peut pas être modifié.
7. Choisissez Enregistrer.

Pour créer une alerte d'utilisation des données par groupe de travail

Vous pouvez définir plusieurs seuils d'alerte lorsque des requêtes exécutées dans un groupe de travail analysent une quantité de données spécifiée au cours d'une période donnée. Les alertes sont mises en œuvre à l'aide des CloudWatch alarmes Amazon et s'appliquent à toutes les requêtes du groupe de travail. Lorsqu'un seuil est atteint, Amazon SNS peut envoyer un e-mail aux utilisateurs que vous spécifiez. Les requêtes ne sont pas automatiquement annulées lorsqu'un seuil est atteint.

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Dans le panneau de navigation, choisissez Groupes de travail.
4. Choisissez le nom du groupe de travail dans la liste.
5. Choisissez Edit (Modifier) pour modifier les paramètres du groupe de travail.
6. Faites défiler l'écran jusqu'à et développer Workgroup data usage alerts – optional (Alertes d'utilisation des données de groupe de travail – facultatif).
7. Choisissez Add alert (Ajouter une alerte).
8. Pour Data usage threshold configuration (Configuration du seuil d'utilisation des données), spécifiez les valeurs comme suit :
   - Pour Data threshold (Seuil de données), spécifiez un nombre, puis sélectionnez une valeur unitaire dans la liste déroulante.
   - Pour Time period (Période), choisissez une période dans la liste déroulante.
   - Pour SNS topic selection (Sélection des rubriques SNS), choisissez une rubrique Amazon SNS dans la liste déroulante. Vous pouvez aussi choisir Create SNS topic (Créer une rubrique SNS) pour accéder directement à la Console Amazon SNS, créer la rubrique Amazon SNS et configurer un abonnement pour l'un des utilisateurs de votre compte Athena. Pour plus d'informations, consultez Prise en main d'Amazon SNS dans le Guide du développeur Amazon Simple Notification Service.
9. Choisissez Add alert (Ajouter une alerte) si vous créez une alerte, ou Save (Enregistrer) pour enregistrer une alerte existante.

Gestion de la capacité de traitement des requêtes

Vous pouvez utiliser des réserves de capacité pour spécifier une capacité de traitement dédiée pour les requêtes que vous exécutez dans Athena. Grâce aux réserves de capacité, vous pouvez tirer parti des capacités de gestion des charges de travail qui vous aident à hiérarchiser, contrôler et mettre à l'échelle vos charges de travail interactives les plus importantes. Par exemple, vous pouvez ajouter une capacité à tout moment pour augmenter le nombre de requêtes que vous pouvez exécuter simultanément, contrôler les charges de travail pouvant utiliser cette capacité et partager la capacité entre les charges de travail. La capacité est entièrement gérée par Athena et détenue pour vous aussi longtemps que vous le souhaitez. La configuration est simple et aucune modification de vos instructions SQL n'est requise.

Pour obtenir une capacité de traitement pour vos requêtes, vous créez une réserve de capacité, vous spécifiez le nombre d'unités de traitement des données (DPU) dont vous avez besoin et vous attribuez un ou plusieurs groupes de travail à la réserve.

Les groupes de travail jouent un rôle important lorsque vous utilisez les réserves de capacité. Les groupes de travail vous permettent d'organiser les requêtes en regroupements logiques. Grâce aux réserves de capacité, vous attribuez une capacité de manière sélective aux groupes de travail afin de contrôler le comportement des requêtes pour chaque groupe de travail et la manière dont elles sont facturées. Pour plus d'informations sur les groupes de travail, consultez Utilisation des groupes de travail pour contrôler l'accès aux requêtes et les coûts.

L'attribution de groupes de travail aux réserves vous permet de donner la priorité aux requêtes que vous soumettez aux groupes de travail attribués. Par exemple, vous pouvez allouer une capacité à un groupe de travail utilisé pour les requêtes d'information financière urgentes afin d'isoler ces requêtes des requêtes moins critiques d'un autre groupe de travail. Cela permet une exécution cohérente des requêtes pour les charges de travail critiques tout en permettant à d'autres charges de travail de s'exécuter indépendamment.

Vous pouvez utiliser les réserves de capacité et les groupes de travail ensemble pour répondre à différentes exigences. Voici des exemples de scénarios :

- Isolation : pour isoler une charge de travail importante, vous attribuez un seul groupe de travail à une réserve. Seules les requêtes du groupe de travail désigné utilisent la capacité de traitement de la réserve choisie.
- Partage : plusieurs charges de travail peuvent partager la capacité d'une seule réserve. Par exemple, si vous souhaitez un coût mensuel prévisible pour un ensemble spécifique de charges

de travail, vous pouvez attribuer plusieurs groupes de travail à une seule réserve. Les groupes de travail attribués partagent la capacité de la réserve.

- Modèle mixte : vous pouvez utiliser les réserves de capacité et la facturation par requête en même temps dans le même compte. Par exemple, pour garantir l'exécution fiable des requêtes prenant en charge une application de production, vous attribuez un groupe de travail chargé de ces requêtes à une réserve de capacité. Lorsque vous développez les requêtes avant de les déplacer vers le groupe de travail de production, vous utilisez un groupe de travail distinct qui n'est pas associé à une réserve et qui utilise donc la facturation par requête.

Compréhension des DPU

La capacité est mesurée en unités de traitement de données (DPU). Les DPU représentent les ressources de calcul et de mémoire utilisées par Athena pour accéder aux données et les traiter en votre nom. Une DPU fournit 4 vCPU et 16 Go de mémoire. Le nombre de DPU que vous spécifiez influence le nombre de requêtes que vous pouvez exécuter simultanément. Par exemple, une réserve comprenant 256 DPU peut prendre en charge environ deux fois plus de requêtes simultanées qu'une réserve comprenant 128 DPU.

Vous pouvez créer jusqu'à 100 réserves de capacité comprenant un maximum de 1 000 DPU au total par compte et par région. Le nombre minimum de DPU que vous pouvez demander est de 24. Si vous avez besoin de plus de 1 000 DPU pour votre cas d'utilisation, contactez athena-feedback@amazon.com.

Pour plus d'informations sur l'estimation de vos exigences de capacité, consultezDétermination des exigences de capacité. Pour de plus amples informations, consultez la rubrique Tarification Amazon Athena.

Considérations et restrictions

- La fonctionnalité nécessite la version 3 du moteur Athena.
- Vous ne pouvez attribuer qu'un seul groupe de travail tout au plus à une réserve à la fois et vous pouvez ajouter un maximum de 20 groupes de travail à une réserve.
- Vous ne pouvez pas ajouter de groupes de travail compatibles avec Spark à une réserve de capacité.
- Pour supprimer un groupe de travail attribué à une réserve, supprimez d'abord le groupe de travail de la réserve.
- Le nombre minimum de DPU que vous pouvez allouer est de 24.

- Vous pouvez créer jusqu'à 100 réserves de capacité comprenant un maximum de 1 000 DPU au total par compte et par région.
- Les demandes de capacité ne sont pas garanties et peuvent prendre jusqu'à 30 minutes.
- La période de facturation minimale est d'une heure par réserve. Après une heure, la capacité est facturée à la minute. Pour de plus amples informations, consultez la rubrique Tarification Amazon Athena.
- La capacité réservée n'est pas transférable à une autre réserve de capacité, un autre Compte AWS ou une autre Région AWS.
- Les requêtes DDL sur les réserves de capacité consomment des DPU.
- Les requêtes exécutées sur la capacité allouée ne sont pas prises en compte dans vos limites de requêtes actives pour DDL et DML.
- Si toutes les DPU sont utilisées, les requêtes soumises sont mises en file d'attente. Ces demandes ne sont pas rejetées et ne sont pas affectées à la capacité à la demande.
- La `DPUConsumed` CloudWatch métrique est par groupe de travail plutôt que par réservation. Si vous déplacez un groupe de travail d'une réserve à une autre, la métrique `DPUConsumed` inclut les données de la période pendant laquelle le groupe de travail appartenait à la première réserve. Pour plus d'informations sur l'utilisation CloudWatch des métriques dans Athena, consultez. Surveillance des requêtes Athena à l'aide de métriques CloudWatch
- Actuellement, cette fonctionnalité est disponible dans les versions suivantes Régions AWS :
  - USA Est (Virginie du Nord)
  - USA Est (Ohio)
  - US West (Oregon)
  - Asie-Pacifique (Singapour)
  - Asie-Pacifique (Sydney)
  - Asia Pacific (Tokyo)
  - Europe (Ireland)
  - Europe (Espagne)
  - Europe (Stockholm)
  - Amérique du Sud (São Paulo)

Rubriques

- Détermination des exigences de capacité

- Création d'une réserve de capacité
- Gestion des réserves
- Politiques IAM pour les réserves de capacité
- API de réserve de capacité Athena

Détermination des exigences de capacité

Avant de créer une réserve de capacité, vous pouvez estimer la capacité requise afin de pouvoir lui attribuer le nombre correct de DPU. Ensuite, une fois qu'une réserve est en cours d'utilisation, vous souhaiterez peut-être vérifier si sa capacité est insuffisante ou excédentaire. Cette rubrique décrit les techniques que vous pouvez utiliser pour réaliser ces estimations et décrit également certains AWS outils permettant d'évaluer l'utilisation et les coûts.

Rubriques

- Estimation de la capacité requise
- Signes indiquant qu'une capacité accrue est requise
- Vérification de la capacité inutilisée
- Outils d'évaluation des exigences de capacité et des coûts

Estimation de la capacité requise

Lors de l'estimation des exigences de capacité, il est utile de prendre en compte deux points de vue : la capacité dont une requête particulière peut avoir besoin et la capacité dont vous pourriez avoir besoin en général.

Estimation des exigences de capacité par requête

Pour déterminer le nombre de DPU qu'une requête peut nécessiter, vous pouvez suivre les directives suivantes :

- Les requêtes DDL consomment 4 DPU.
- Les requêtes DML consomment généralement entre 4 et 124 DPU.

Athena détermine le nombre de DPU requis par une requête DML lorsque celle-ci est soumise. Le nombre varie en fonction de la taille des données, du format de stockage, de la construction de

la requête et d'autres facteurs. En général, Athena essaie de sélectionner le nombre de DPU le plus bas et le plus efficace. Si Athena détermine qu'une puissance de calcul plus importante est nécessaire pour que la requête soit menée à bien, elle augmente le nombre de DPU attribués à la requête.

Estimation des exigences de capacité spécifiques à la charge de travail

Pour déterminer la capacité dont vous pourriez avoir besoin pour exécuter plusieurs requêtes en même temps, prenez en compte les directives générales du tableau suivant :

Requêtes simultanées	DPU requises
10	40 ou plus
20	96 ou plus
30 ou plus	240 ou plus

Notez que le nombre réel de DPU dont vous avez besoin dépend de vos objectifs et de vos modèles d'analyse. Par exemple, si vous souhaitez que les requêtes démarrent immédiatement sans mise en file d'attente, déterminez votre demande maximale de requêtes simultanées, puis allouez le nombre de DPU en conséquence.

Vous pouvez allouer moins de DPU que votre pic de demande, mais une mise en file d'attente peut se produire en cas de pic de demande. Lors de la mise en file d'attente, Athena place vos requêtes dans une file d'attente et les exécute lorsque la capacité devient disponible.

Si votre objectif est d'exécuter des requêtes dans les limites d'un budget fixe, vous pouvez utiliser le Calculateur de prix AWS pour déterminer le nombre de DPU requises.

Enfin, n'oubliez pas que la taille des données, le format de stockage et la manière dont une requête est écrite influencent les DPU requises par une requête. Pour améliorer les performances des requêtes, vous pouvez compresser ou partitionner vos données ou les convertir en formats en colonnes. Pour plus d'informations, consultez Réglage de performances dans Athena.

Signes indiquant qu'une capacité accrue est requise

Les messages d'erreur relatifs à une capacité insuffisante et la mise en file d'attente des requêtes indiquent que la capacité qui vous est attribuée est inadéquate.

Si vos requêtes échouent avec un message d'erreur indiquant une capacité insuffisante, c'est que le nombre de DPU de votre réserve de capacité est trop faible pour votre requête. Par exemple, si vous avez une réserve comprenant 24 DPU et que vous exécutez une requête qui requiert plus de 24 DPU, la requête échouera. Pour détecter cette erreur de requête, vous pouvez utiliser les EventBridgeévénements d'Athéna. Essayez d'ajouter des DPU et de réexécuter votre requête.

Si de nombreuses requêtes sont mises en file d'attente, cela signifie que votre capacité est pleinement utilisée par d'autres requêtes. Pour réduire la mise en file d'attente, effectuez l'une des actions suivantes :

- Ajouter des DPU à votre réserve pour augmenter la simultanéité des requêtes.
- Supprimer des groupes de travail de votre réserve afin de libérer de la capacité pour d'autres requêtes.

Pour vérifier l'absence de files d'attente excessives, utilisez l'CloudWatchindicateur de temps de file d'attente des requêtes Athena pour les groupes de travail inclus dans votre réservation de capacité. Si la valeur est supérieure à votre seuil préféré, vous pouvez ajouter des DPU à la réserve de capacité.

Vérification de la capacité inutilisée

Pour vérifier la capacité inutilisée, vous pouvez soit diminuer le nombre de DPU dans la réserve, soit augmenter sa charge de travail, puis observer les résultats.

Pour vérifier la capacité inutilisée

1. Effectuez l’une des actions suivantes :
   - Réduire le nombre de DPU de votre réserve (réduire les ressources disponibles)
   - Ajouter des groupes de travail à votre réserve (augmenter la charge de travail)
2. CloudWatchÀ utiliser pour mesurer le temps de file d'attente des requêtes.
3. Si le temps de file d'attente augmente au-delà d'un niveau souhaitable, effectuez l'une des actions suivantes :
   - Supprimer des groupes de travail
   - Ajouter des DPU à votre réserve de capacité
4. Après chaque modification, vérifiez les performances et le temps de file d'attente des requêtes.

5. Continuez à ajuster la charge de travail et/ou le nombre de DPU pour atteindre l'équilibre souhaité.

Si vous ne souhaitez pas maintenir la capacité en dehors d'une période préférée, vous pouvez annuler la réserve et en créer une autre ultérieurement. Toutefois, même si vous avez récemment annulé la capacité d'une autre réserve, les demandes de nouvelles capacités ne sont pas garanties et la création de nouvelles réserves prend du temps.

Outils d'évaluation des exigences de capacité et des coûts

Vous pouvez utiliser les services et fonctionnalités suivants AWS pour mesurer votre utilisation et vos coûts d'Athena.

CloudWatchmétriques

Vous pouvez configurer Athena pour publier les métriques liées aux requêtes sur Amazon CloudWatch au niveau du groupe de travail. Une fois que vous avez activé les métriques pour le groupe de travail, les métriques pour les requêtes du groupe de travail s'affichent dans la console Athena sur la page de détails du groupe de travail.

Pour plus d'informations sur les métriques Athena publiées sur CloudWatch et leurs dimensions, consultez. Surveillance des requêtes Athena à l'aide de métriques CloudWatch

CloudWatch métriques d'utilisation

Vous pouvez utiliser les statistiques CloudWatch d'utilisation pour avoir une idée de la manière dont votre compte utilise les ressources en affichant votre utilisation actuelle des services sur CloudWatch des graphiques et des tableaux de bord. Pour Athena, les mesures de disponibilité d'utilisation correspondent aux quotas de AWS service pour Athena. Vous pouvez configurer des alarmes qui vous alertent lorsque votre utilisation approche d’un quota de service.

Pour plus d’informations, consultez Surveillance des métriques d'utilisation d'Athena.

EventBridgeÉvénements Amazon

Vous pouvez utiliser Amazon Athena avec Amazon EventBridge pour recevoir des notifications en temps réel concernant l'état de vos requêtes. Lorsqu'une requête que vous avez soumise change d'état, Athena publie un événement EventBridge contenant des informations sur le changement d'état de la requête. Vous pouvez écrire des règles simples pour les événements qui vous intéressent et effectuer des actions automatisées lorsqu'un événement correspond à une règle.

Pour plus d'informations, veuillez consulter les ressources suivantes.

- Surveillance des requêtes Athena à l'aide des événements Amazon EventBridge
- Qu'est-ce qu'Amazon EventBridge ?
- EventBridge Événements Amazon

Balises

Dans Athena, les réserves de capacité prennent en charge les balises. Une balise se compose d'une clé et d'une valeur. Pour suivre vos coûts dans Athena, vous pouvez utiliser des balises de AWS répartition des coûts générées. AWS utilise les balises de répartition des coûts pour organiser les coûts des ressources dans votre rapport sur les coûts et l'utilisation. Cela vous permet de classer et de suivre plus facilement vos coûts AWS . Pour activer les balises de répartition des coûts pour Athena, vous devez utiliser la console AWS Billing and Cost Management.

Pour plus d'informations, veuillez consulter les ressources suivantes.

- Étiquetage des ressources Athena
- Activation des AWS balises de répartition des coûts générées
- Utilisation des balises de répartition des coûts AWS

Création d'une réserve de capacité

Pour commencer, vous créez une réserve de capacité comportant le nombre de DPU dont vous avez besoin, puis vous attribuez un ou plusieurs groupes de travail qui utiliseront cette capacité pour leurs requêtes. Vous pouvez ajuster votre capacité ultérieurement si nécessaire pour fournir des performances plus constantes ou mieux gérer les coûts. Pour plus d'informations sur l'estimation de vos exigences de capacité, consultezDétermination des exigences de capacité.

> **Important**
>
> Les demandes de capacité ne sont pas garanties et peuvent prendre jusqu'à 30 minutes.

Pour créer une réserve de capacité

1. Ouvrez la console Athena à l’adresse https://console.aws.amazon.com/athena/.

2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Choisissez Administration, Réserves de capacité.
4. Choisissez Créer une réserve de capacité.
5. Sur la page Créer une réserve de capacité, saisissez le nom dans le champ Nom de la réserve de capacité. Le nom doit être unique, compter de 1 à 128 caractères, et utiliser uniquement les caractères a-z, A-Z, 0-9, _ (trait de soulignement), . (point) et - (trait d'union). Vous ne pouvez pas modifier le nom une fois la réserve créée.
6. Dans DPU, choisissez ou saisissez le nombre d'unités de traitement de données (DPU) que vous souhaitez, par incréments de 4. Pour plus d'informations, consultez Compréhension des DPU.
7. (Facultatif) Développez l'option Balises, puis choisissez Ajouter une nouvelle balise pour ajouter une ou plusieurs paires clé/valeur personnalisées à associer à la ressource de réserve de capacité. Pour plus d'informations, consultez Étiquetage des ressources Athena.
8. Choisissez Examiner.
9. À l'invite Confirmer la réservation de capacité, confirmez le nombre de DPU et Région AWS d'autres informations. Si vous acceptez, choisissez Soumettre.

   Sur la page de détails, l'état de votre réserve de capacité indique En attente. Lorsque votre réserve de capacité est disponible pour exécuter des requêtes, son état s'affiche comme Actif.

À ce stade, vous êtes prêt à ajouter un ou plusieurs groupes de travail à votre réserve. Pour les étapes, consultez Ajout de groupes de travail à une réserve.

Gestion des réserves

Vous pouvez consulter et gérer vos réserves de capacité sur la page Réserves de capacité. Vous pouvez effectuer des tâches de gestion telles que l'ajout ou la réduction des DPU, la modification des attributions des groupes de travail et le balisage ou l'annulation de réserves.

Pour consulter et gérer des réserves de capacité

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Choisissez Administration, Réserves de capacité.

4. Vous pouvez effectuer les tâches suivantes sur la page des réserves de capacité :

   - Pour créer une réserve de capacité, choisissez Créer une réserve de capacité.
   - Utilisez le champ de recherche pour filtrer les réserves par nom ou par nombre de DPU.
   - Choisissez le menu déroulant d'état pour filtrer par état de réserve de capacité (par exemple, Actif ou Annulé). Pour de plus amples informations sur l'état des réserves, consultez Compréhension de l'état des réserves.
   - Pour consulter les détails d'une réserve de capacité, cliquez sur le lien correspondant à la réserve. La page de détails de la réserve inclut des options relatives à la modification de la capacité, à l'ajout de groupes de travail, à la suppression de groupes de travail et à l'annulation de la réserve.
   - Pour modifier une réserve (par exemple, en ajoutant ou en supprimant des DPU), sélectionnez le bouton correspondant à la réserve, puis choisissez Modifier.
   - Pour annuler une réserve, sélectionnez le bouton correspondant à la réserve, puis cliquez sur Annuler.

Compréhension de l'état des réserves

Le tableau suivant décrit les valeurs d'état possibles pour une réserve de capacité.

État	Description
En suspens	Athena est en train de traiter votre demande de capacité. La capacité n'est pas prête à exécuter des requêtes.
Actif	La capacité est disponible pour exécuter des requêtes.
Échec	Votre demande de capacité n'a pas été traitée avec succès. Notez que le traitement des demandes de capacité n'est pas garanti. Les réserves qui ont échoué sont prises en compte dans le calcul des limites de DPU de votre compte. Pour libérer l'utilisation, vous devez annuler la réserve.
Mise à jour en attente	Athena est en train de modifier la réserve. Par exemple, cet état apparaît une fois que vous avez modifié la réserve pour ajouter ou supprimer des DPU.

État	Description
Annulation	Athena est en train de traiter une demande d'annulation de réserve. Les requêtes toujours en cours d'exécution dans les groupes de travail qui utilisaient la réserve sont autorisées à se terminer, mais les autres requêtes du groupe de travail utiliseront la capacité à la demande (non allouée).
Annulée	L'annulation de la réserve de capacité est terminée. Les réserves annulées restent dans la console pendant 45 jours. Après 45 jours, Athena annulera la réserve. Pendant les 45 jours, vous ne pouvez pas réaffecter ou réutiliser la réserve, mais vous pouvez vous référer à ses balises et consulter ses détails pour une référence historique. Il n'est pas garanti que la capacité annulée puisse être réservée à nouveau à une date future. La capacité ne peut pas être transférée à une autre réservation, Compte AWS ou Région AWS.

Comprendre les DPU actives et les DPU cibles

Dans la liste des réserves de capacité de la console Athena, votre réserve affiche deux valeurs DPU : DPU active et DPU cible.

- DPU active : nombre de DPU disponibles dans votre réserve pour exécuter des requêtes. Par exemple, si vous demandez 100 DPU et que votre demande est satisfaite, DPU active affiche 100.
- DPU cible : nombre de DPU vers lesquelles votre réserve est en cours de déplacement. DPU cible affiche une valeur différente de celle de DPU active lorsqu'une réserve est créée ou lorsqu'une augmentation ou une diminution du nombre de DPU est en attente.

Par exemple, après avoir soumis une demande de création d'une réserve comprenant 24 DPU, l'état de la réserve sera En attente, DPU active sera 0 et DPU cible sera 24.

Si vous avez une réserve comprenant 100 DPU et que vous modifiez celle-ci pour demander une augmentation de 20 DPU, l'état sera Mise à jour en attente, DPU active sera de 100 et DPU cible de 120.

Si vous avez une réserve comprenant 100 DPU et que vous modifiez celle-ci pour demander une diminution de 20 DPU, l'état sera Mise à jour en attente, DPU active sera de 100 et DPU cible de 80.

Au cours de ces transitions, Athena travaille activement à l'acquisition ou à la réduction du nombre de DPU en fonction de votre demande. Lorsque DPU active devient égal à DPU cible, le nombre cible est atteint et aucune modification n'est en attente.

Pour récupérer ces valeurs par programmation, vous pouvez appeler l'action GetCapacityReservationAPI. L'API fait référence à DPU active et DPU cible sous `AllocatedDpus` et `TargetDpus`.

Rubriques

- Modification de réserves de capacité
- Ajout de groupes de travail à une réserve
- Suppression d'un groupe de travail d'une réserve
- Annulation d'une réserve de capacité
- Suppression d'une réserve de capacité

Modification de réserves de capacité

Après avoir créé une réserve de capacité, vous pouvez ajuster son nombre de DPU et ajouter ou supprimer ses balises personnalisées.

Pour modifier une réserve de capacité

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Choisissez Administration, Réserves de capacité.
4. Dans la liste des réserves de capacité, effectuez l'une des opérations suivantes :
   - Sélectionner le bouton en regard de la réserve, puis choisir Modifier.
   - Choisir le lien de la réserve, puis choisir Modifier.
5. Dans DPU, choisissez ou saisissez le nombre d'unités de traitement de données que vous souhaitez, par incréments de 4. Le nombre minimum de DPU que vous pouvez avoir est de 24. Pour plus d'informations, consultez Compréhension des DPU.

> **Note**
>
> Vous pouvez ajouter des DPU à une réserve de capacité existante à tout moment. Toutefois, vous ne pouvez réduire le nombre de DPU qu'au plus tôt une heure après avoir créé la réservation ou y avoir ajouté des DPU.

6. (Facultatif) Dans Balises, choisissez Supprimer la balise pour supprimer une balise ou Ajouter une balise pour ajouter une nouvelle balise.
7. Sélectionnez Envoyer. La page de détails de la réserve affiche la configuration mise à jour.

Ajout de groupes de travail à une réserve

Après avoir créé une réserve de capacité, vous pouvez ajouter jusqu'à 20 groupes de travail à la réserve. L'ajout d'un groupe de travail à une réserve indique à Athena quelles requêtes doivent être exécutées sur la capacité réservée. Les requêtes provenant de groupes de travail qui ne sont pas associées à une réserve continuent d'être exécutées selon le modèle de tarification par téraoctet (To) analysé par défaut.

Lorsqu'une réserve comporte deux groupes de travail ou plus, les requêtes provenant de ces groupes de travail peuvent utiliser la capacité de la réserve. Vous pouvez ajouter et supprimer des groupes de travail à tout moment. Lorsque vous ajoutez ou supprimez des groupes de travail, les requêtes en cours d'exécution ne sont pas interrompues.

Lorsque votre réserve est en attente, les requêtes provenant des groupes de travail que vous avez ajoutés continuent de s'exécuter en utilisant le modèle de tarification par téraoctet (To) analysé par défaut jusqu'à ce que la réserve soit active.

Pour ajouter un ou plusieurs groupes de travail à votre réserve de capacité

1. Sur la page de détails de la réserve de capacité, choisissez Ajouter des groupes de travail.
2. Sur la page Ajouter des groupes de travail, sélectionnez les groupes de travail que vous souhaitez ajouter, puis choisissez Ajouter des groupes de travail. Vous pouvez attribuer un groupe de travail à plusieurs réserves.

   La page de détails de votre réserve de capacité répertorie les groupes de travail que vous avez ajoutés. Les requêtes exécutées dans ces groupes de travail utiliseront la capacité que vous avez réservée lorsque la réserve est active.

Suppression d'un groupe de travail d'une réserve

Si vous n'avez plus besoin de capacité dédiée pour un groupe de travail ou si vous souhaitez déplacer un groupe de travail vers sa propre réserve, vous pouvez le supprimer à tout moment. La suppression d'un groupe de travail d'une réserve est un processus simple. Une fois que vous avez supprimé un groupe de travail d'une réserve, les requêtes du groupe de travail supprimé utilisent par défaut une capacité à la demande (non allouée) et sont facturées sur la base des téraoctets (To) analysés.

Pour supprimer un ou plusieurs groupes de travail d'une réserve

1. Sur la page de détails de la réserve de capacité, sélectionnez les groupes de travail que vous souhaitez supprimer.
2. Choisissez Supprimer les groupes de travail. L'invite Supprimer les groupes de travail ? vous informe que toutes les requêtes actuellement actives seront terminées avant que le groupe de travail ne soit supprimé de la réserve.
3. Sélectionnez Remove (Supprimer). La page de détails de votre réserve de capacité indique que les groupes de travail supprimés ne sont plus présents.

Annulation d'une réserve de capacité

Si vous ne souhaitez plus utiliser la réserve de capacité, vous pouvez l'annuler. Les requêtes toujours en cours d'exécution dans les groupes de travail qui utilisaient la réserve seront autorisées à se terminer, mais les autres requêtes du groupe de travail n'utiliseront plus la réserve.

> **Note**
>
> Il n'est pas garanti que la capacité annulée puisse être réservée à nouveau à une date future. La capacité ne peut pas être transférée à une autre réservation, Compte AWS ou Région AWS.

Pour annuler une réserve de capacité

1. Ouvrez la console Athena à l’adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Choisissez Administration, Réserves de capacité.

4. Dans la liste des réserves de capacité, effectuez l'une des opérations suivantes :

   - Sélectionner le bouton en regard de la réserve, puis choisir Annuler.
   - Choisir le lien de réserve, puis choisir Annuler la réserve de capacité.

5. À l'invite Annuler la réserve de capacité ?, saisissez Annuler, puis choisissez Annuler la réserve de capacité.

   L'état de la réserve passe à Annulation, et une bannière de progression vous informe que l'annulation est en cours.

   Lorsque l'annulation est terminée, la réserve de capacité est maintenue, mais son état est Annulé. La réserve sera supprimée 45 jours après l'annulation. Pendant les 45 jours, vous ne pouvez pas réaffecter ou réutiliser la réserve annulée, mais vous pouvez vous référer à ses balises et consulter ses détails pour une référence historique.

Suppression d'une réserve de capacité

Si vous souhaitez supprimer toutes les références à une réserve de capacité annulée, vous pouvez supprimer la réserve. La réserve doit être annulée avant de pouvoir être supprimée. Une réserve supprimée est immédiatement supprimée de votre compte et ne peut plus être référencée, y compris par son ARN.

Pour supprimer une réserve de capacité

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Choisissez Administration, Réserves de capacité.
4. Dans la liste des réserves de capacité, effectuez l'une des opérations suivantes :

   - Sélectionner le bouton en regard de la réserve annulée, puis choisir Actions, Supprimer.
   - Choisir le lien de la réserve, puis choisir Supprimer.

5. À l'invite Supprimer la réserve de capacité ?, choisissez Supprimer.

   Une bannière vous informe que la réserve de capacité a été supprimée avec succès. La réserve supprimée n'apparaît plus dans la liste des réserves de capacité.

Politiques IAM pour les réserves de capacité

Pour contrôler l'accès aux réserves de capacité, utilisez des autorisations IAM au niveau des ressources ou des politiques IAM basées sur l'identité. Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

La procédure suivante est spécifique à Athena.

Pour des informations spécifiques à IAM, consultez les liens répertoriés à la fin de cette section. Pour de plus amples informations sur les politiques de réserve de capacité JSON, consultez Exemples de politiques de réserve de capacité.

Utilisation de l'éditeur visuel dans la console IAM pour créer une politique de réserve de capacité

1. Connectez-vous à l'outil AWS Management Console, puis ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation à gauche, choisissez politiques, puis Créer une politique.
3. Dans l'onglet Visual editor (Éditeur visuel), sélectionnez Choose a service (Choisir un service). Choisissez ensuite Athena pour l'ajouter à la politique.
4. Choisissez Sélectionner des actions, puis choisissez les actions à ajouter à la politique. L'éditeur visuel affiche les actions disponibles dans Athena. Pour plus d'informations, consultez la rubrique Actions, ressources et clés de condition pour Amazon Athena dans la section Référence de l'autorisation de service.
5. Choisissez Ajouter des actions pour entrer une action spécifique ou utilisez des caractères génériques (*) pour spécifier plusieurs actions.

   Par défaut, la politique que vous créez autorise les actions que vous choisissez. Si vous avez choisi une ou plusieurs actions qui prennent en charge les autorisations au niveau des ressources pour la ressource `capacity-reservation` dans Athena, l'éditeur visuel affiche la ressource `capacity-reservation`.
6. Choisissez Ressources pour spécifier les réserves de capacité spécifiques de votre politique. Pour un exemple de politiques de réserve de capacité JSON, consultez Exemples de politiques de réserve de capacité.
7. Spécifiez la ressource du `capacity-reservation` comme suit :

```
arn:aws:athena:<region>:<user-account>:capacity-reservation/<capacity-reservation-name>
```

8. Choisissez Review policy (Examiner une politique), puis saisissez un Name (Nom) et une Description (facultatif) pour la politique que vous êtes en train de créer. Passez en revue le résumé de politique afin de vous assurer que les autorisations nécessaires vous ont été accordées.
9. Choisissez Create policy (Créer une politique) pour enregistrer votre nouvelle politique.
10. Attachez cette politique basée sur l'identité à un utilisateur, un groupe ou un rôle.

Pour plus d'informations, consultez les rubriques suivantes dans la Référence de l'autorisation de service et le Guide de l'utilisateur IAM :

- Actions, ressources et clés de condition pour Amazon Athena
- Création de politiques avec l'éditeur visuel
- Ajout et suppression de politiques IAM
- Contrôle de l'accès aux ressources

Pour un exemple de politiques de réserve de capacité JSON, consultez Exemples de politiques de réserve de capacité.

Pour obtenir une liste complète d'actions Amazon Athena, consultez les noms d'action API dans la rubrique Référence d'API Amazon Athena.

Exemples de politiques de réserve de capacité

Cette section inclut des exemples de politiques que vous pouvez utiliser pour activer plusieurs actions sur des réserves de capacité. Chaque fois que vous utilisez des politiques IAM, veillez à respecter les bonnes pratiques IAM. Pour plus d'informations, consultez la rubrique Bonnes pratiques IAM du Guide de l'utilisateur IAM.

Une réserve de capacité est une ressource IAM gérée par Athena. Par conséquent, si votre politique de réserve de capacité utilise des actions prenant `capacity-reservation` en entrée, vous devez spécifier l'ARN de la réserve de capacité comme suit :

```
"Resource": [arn:aws:athena:<region>:<user-account>:capacity-reservation/<capacity-reservation-name>]
```

Où *<capacity-reservation-name>* est le nom de votre réserve de capacité. Par exemple, pour une réserve de capacité nommée `test_capacity_reservation`, spécifiez-la en tant que ressource comme suit :

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:capacity-reservation/
test_capacity_reservation"]
```

Pour obtenir une liste complète d'actions Amazon Athena, consultez les noms d'action d'API dans la Référence d'API Amazon Athena. Pour plus d'informations sur les politiques IAM, consultez la rubrique Création de politiques avec l'éditeur visuel du Guide de l'utilisateur IAM.

- Example policy to list capacity reservations
- Example policy for management operations

Example Exemple de politique pour répertorier les réserves de capacité

La politique suivante permet à tous les utilisateurs de répertorier toutes les réserves de capacité.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListCapacityReservations"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Exemple de politique pour les opérations de gestion

La politique suivante permet à un utilisateur de créer, d'annuler, d'obtenir des informations sur et de mettre à jour la réserve de capacité `test_capacity_reservation`. La politique permet également à un utilisateur d'attribuer les `workgroupA` et `workgroupB` à la `test_capacity_reservation`.

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

```
      {
         "Effect": "Allow",
         "Action": [
             "athena:CreateCapacityReservation",
             "athena:GetCapacityReservation",
             "athena:CancelCapacityReservation",
             "athena:UpdateCapacityReservation",
             "athena:GetCapacityAssignmentConfiguration",
             "athena:PutCapacityAssignmentConfiguration"
         ],
         "Resource": [
             "arn:aws:athena:us-east-1:123456789012:capacity-
reservation/test_capacity_reservation",
             "arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA",
             "arn:aws:athena:us-east-1:123456789012:workgroup/workgroupB"
         ]
      }
   ]
}
```

API de réserve de capacité Athena

La liste suivante contient des liens de référence vers les actions des API de réserve de capacité Athena. Pour les structures de données et les autres actions des API Athena, voir la Référence d'API Amazon Athena.

- CancelCapacityReservation
- CreateCapacityReservation
- GetCapacityAssignmentConfiguration
- GetCapacityReservation
- ListCapacityReservations
- PutCapacityAssignmentConfiguration
- UpdateCapacityReservation

Réglage de performances dans Athena

Cette rubrique fournit des informations générales et des suggestions spécifiques pour améliorer les performances de vos requêtes Athena, et explique comment contourner les erreurs liées aux limites et à l'utilisation des ressources.

Quotas de service

Athena applique des quotas pour les métriques telles que la durée d'exécution des requêtes, le nombre de requêtes simultanées dans un compte et les taux de demandes d'API. Pour de plus amples informations sur ces quotas, consultez Service Quotas. Le dépassement de ces quotas entraîne l'échec d'une requête, soit lors de son envoi, soit lors de son exécution.

La plupart des conseils d'optimisation des performances présentés sur cette page peuvent contribuer à réduire la durée d'exécution des requêtes. L'optimisation libère de la capacité afin que vous puissiez exécuter davantage de requêtes dans les limites du quota de simultanéité et évite que les requêtes ne soient annulées pour cause de trop longue durée.

Les quotas relatifs au nombre de requêtes simultanées et de demandes d'API sont calculés par Compte AWS et Région AWS. Nous recommandons d'exécuter une charge de travail par charge de travail Compte AWS (ou d'utiliser des réservations de capacité provisionnées distinctes) pour éviter que les charges de travail ne soient en concurrence pour le même quota.

Si vous exécutez deux charges de travail dans le même compte, l'une des charges de travail peut exécuter une rafale de requêtes. Cela peut entraîner une limitation de la charge de travail restante ou l'empêcher d'exécuter des requêtes. Pour éviter cela, vous pouvez déplacer les charges de travail vers des comptes distincts afin de donner à chaque charge de travail son propre quota de simultanéité. La création d'une réserve de capacité allouée pour l'une ou les deux charges de travail permet d'atteindre le même objectif.

Quotas dans d'autres services

Lorsqu'Athena exécute une requête, il peut appeler d'autres services qui appliquent des quotas. Pendant l'exécution des requêtes, Athena peut effectuer des appels d'API vers Amazon S3 et d'autres AWS services tels que IAM et. AWS Glue Data Catalog AWS KMS Si vous utilisez des requêtes fédérées, Athena appelle également. AWS Lambda Tous ces services ont leurs propres limites et quotas qui peuvent être dépassés. Lorsque l'exécution d'une requête rencontre des erreurs provenant de ces services, elle échoue et inclut l'erreur provenant du service source. Les erreurs récupérables font l'objet de nouvelles tentatives, mais les requêtes peuvent toujours échouer si le problème ne se résout pas de lui-même à temps. Assurez-vous de lire attentivement les messages d'erreur afin de déterminer s'ils proviennent d'Athena ou d'un autre service. Certaines des erreurs pertinentes sont abordées dans ce document.

Pour plus d'informations sur la manière de contourner les erreurs causées par les Service Quotas d'Amazon S3, consultez Éviter d'avoir trop de fichiers ultérieurement dans ce document. Pour de

plus amples informations sur l'optimisation des performances Amazon S3, consultez Schémas de conception des bonnes pratiques : optimisation des performances Amazon S3 dans le Guide de l'utilisateur Amazon S3.

Limites des ressources

Athena exécute des requêtes dans un moteur de requête distribué. Lorsque vous soumettez une requête, le planificateur de requêtes du moteur Athena estime la capacité de calcul requise pour exécuter la requête et prépare un cluster de nœuds de calcul en conséquence. Certaines requêtes, telles que les requêtes DDL, ne s'exécutent que sur un seul nœud. Les requêtes complexes portant sur des jeux de données volumineux s'exécutent sur des clusters beaucoup plus importants. Les nœuds sont uniformes, avec les mêmes configurations de mémoire, d'UC et de disque. Athena monte en puissance, mais n'augmente pas en capacité, pour traiter des requêtes plus exigeantes.

Parfois, les exigences d'une requête dépassent les ressources disponibles pour le cluster exécutant la requête. Dans ce cas, la requête échoue avec le message d'erreur La requête a épuisé les ressources à ce facteur d'échelle.

La ressource la plus souvent épuisée est la mémoire, mais dans de rares cas, il peut également s'agir d'espace disque. Les erreurs de mémoire se produisent généralement lorsque le moteur exécute une fonction de jointure ou de fenêtrage, mais elles peuvent également se produire lors de comptages et d'agrégations distincts.

Même si une requête échoue une fois avec une erreur « manque de ressources », elle peut réussir lorsque vous l'exécutez à nouveau. L'exécution des requêtes n'est pas déterministe. Des facteurs tels que le temps nécessaire au chargement des données et la manière dont les jeux de données intermédiaires sont répartis sur les nœuds peuvent entraîner une utilisation différente des ressources. Par exemple, imaginez une requête qui joint deux tables et dont la distribution des valeurs pour la condition de jointure est fortement asymétrique. Une telle requête peut réussir la plupart du temps, mais échouer parfois lorsque les valeurs les plus courantes finissent par être traitées par le même nœud.

Pour éviter que vos requêtes n'excèdent les ressources disponibles, suivez les conseils de réglage des performances mentionnés dans ce document. En particulier, pour obtenir des conseils sur la manière d'optimiser les requêtes qui épuisent les ressources disponibles, consultez Optimisation des jointures, Optimisation des fonctions de fenêtrage et Optimisation des requêtes à l'aide d'approximations.

Techniques d'optimisation des requêtes

Utilisez les techniques d'optimisation des requêtes décrites dans cette section pour accélérer l'exécution des requêtes ou pour contourner les requêtes qui dépassent les limites de ressources dans Athena.

Optimisation des jointures

Il existe de nombreuses stratégies différentes pour exécuter des jointures dans un moteur de requête distribué. Les deux plus courantes sont les jointures par hachage distribuées et les requêtes comportant des conditions de jointure complexes.

Jointure par hachage distribuée

Le type de jointure le plus courant utilise une comparaison d'égalité comme condition de jointure. Athena exécute ce type de jointure en tant que jointure par hachage distribuée.

Dans une jointure par hachage distribuée, le moteur crée une table de recherche (table de hachage) à partir de l'un des côtés de la jointure. Ce côté est appelé côté build. Les enregistrements du côté build sont répartis entre les nœuds. Chaque nœud crée une table de recherche pour son sous-ensemble. L'autre côté de la jointure, appelé côté sonde, est ensuite diffusé via les nœuds. Les enregistrements du côté sonde sont répartis sur les nœuds de la même manière que du côté build. Cela permet à chaque nœud d'effectuer la jointure en recherchant les enregistrements correspondants dans sa propre table de recherche.

Lorsque les tables de recherche créées à partir du côté build de la jointure ne rentrent pas dans la mémoire, les requêtes peuvent échouer. Même si la taille totale du côté build est inférieure à la mémoire disponible, les requêtes peuvent échouer si la répartition des enregistrements présente une asymétrie importante. Dans un cas extrême, tous les enregistrements peuvent avoir la même valeur pour la condition de jointure et doivent être conservés en mémoire sur un seul nœud. Même une requête moins asymétrique peut échouer si un ensemble de valeurs est envoyé au même nœud et que le total des valeurs dépasse la mémoire disponible. Les nœuds ont la capacité de répartir des enregistrements sur le disque, mais le déversement ralentit l'exécution des requêtes et peut s'avérer insuffisant pour empêcher l'échec de la requête.

Athena tente de réorganiser les jointures pour utiliser la relation la plus grande comme côté sonde, et la plus petite relation comme côté build. Cependant, comme Athena ne gère pas les données contenues dans les tables, il dispose de peu d'informations et doit souvent partir du principe que la première table est la plus grande et la seconde la plus petite.

Lorsque vous écrivez des jointures avec des conditions de jointure basées sur l'égalité, supposez que la table située à gauche du mot-clé `JOIN` correspond au côté sonde et que la table de droite correspond au côté build. Assurez-vous que la bonne table, le côté build, est la plus petite des tables. S'il n'est pas possible de réduire le côté build de la jointure suffisamment pour tenir en mémoire, envisagez d'exécuter plusieurs requêtes qui joignent des sous-ensembles de la table de build.

Autres types de jointure

Les requêtes comportant des conditions de jointure complexes (par exemple, les requêtes qui utilisent des opérateurs `LIKE`, `>` ou autres) sont souvent exigeantes en termes de calcul. Dans le pire des cas, chaque enregistrement d'un côté de la jointure doit être comparé à tous les enregistrements de l'autre côté de la jointure. Comme la durée d'exécution augmente avec le carré du nombre d'enregistrements, ces requêtes risquent de dépasser la durée d'exécution maximale.

Pour savoir à l'avance comment Athena exécutera votre requête, vous pouvez utiliser l'instruction `EXPLAIN`. Pour plus d’informations, consultez Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena et Explication des résultats de l'instruction EXPLAIN d'Athena.

Optimisation des fonctions de fenêtrage

Les fonctions de fenêtrage étant des opérations gourmandes en ressources, elles peuvent ralentir ou même faire échouer les requêtes avec le message La requête a épuisé les ressources à ce facteur d'échelle. Les fonctions de fenêtrage conservent en mémoire tous les enregistrements sur lesquels elles opèrent afin de calculer leur résultat. Lorsque la fenêtre est très grande, la fonction de fenêtrage peut manquer de mémoire.

Pour vous assurer que vos requêtes s'exécutent dans les limites de mémoire disponibles, réduisez la taille des fenêtres sur lesquelles vos fonctions de fenêtrage opèrent. Pour ce faire, vous pouvez ajouter une clause `PARTITIONED BY` ou réduire la portée des clauses de partitionnement existantes.

Utiliser des fonctions autres que de fenêtrage à la place

Parfois, les requêtes comportant des fonctions de fenêtrage peuvent être réécrites sans fonctions de fenêtrage. Par exemple, au lieu d'utiliser `row_number` pour rechercher les meilleurs enregistrements `N`, vous pouvez utiliser `ORDER BY` et `LIMIT`. Au lieu d'utiliser `row_number` ou `rank` pour dédupliquer des enregistrements, vous pouvez utiliser des fonctions d'agrégation telles que max_by, min_by et arbitrary.

Supposons, par exemple, que vous disposiez d'un jeu de données actualisé par un capteur. Le capteur indique régulièrement l'état de la batterie et inclut certaines métadonnées, telles que

l'emplacement. Si vous souhaitez connaître le dernier état de la batterie de chaque capteur et son emplacement, vous pouvez utiliser cette requête :

```
SELECT sensor_id,
       arbitrary(location) AS location,
       max_by(battery_status, updated_at) AS battery_status
FROM sensor_readings
GROUP BY sensor_id
```

Les métadonnées telles que l'emplacement étant les mêmes pour chaque enregistrement, vous pouvez utiliser la fonction `arbitrary` pour sélectionner n'importe quelle valeur dans le groupe.

Pour connaître le dernier état de la batterie, vous pouvez utiliser la fonction `max_by`. La fonction `max_by` sélectionne la valeur d'une colonne dans l'enregistrement où la valeur maximale d'une autre colonne a été trouvée. Dans ce cas, il renvoie l'état de la batterie de l'enregistrement avec l'heure de la dernière mise à jour au sein du groupe. Cette requête s'exécute plus rapidement et utilise moins de mémoire qu'une requête équivalente dotée d'une fonction de fenêtrage.

Optimisation des agrégations

Lorsqu'Athena effectue une agrégation, il répartit les enregistrements entre les composants master en utilisant les colonnes de la clause `GROUP BY`. Pour que la tâche de mise en correspondance des enregistrements avec des groupes soit aussi efficace que possible, les nœuds tentent de conserver les enregistrements en mémoire mais les déversent sur le disque si nécessaire.

Il est également conseillé d'éviter d'inclure des colonnes redondantes dans les clauses `GROUP BY`. Le nombre réduit de colonnes nécessitant moins de mémoire, une requête décrivant un groupe utilisant moins de colonnes est plus efficace. Les colonnes numériques utilisent également moins de mémoire que les chaînes. Par exemple, lorsque vous agrégez un jeu de données qui possède à la fois un ID de catégorie numérique et un nom de catégorie, utilisez uniquement la colonne ID de catégorie dans la clause `GROUP BY`.

Parfois, les requêtes incluent des colonnes dans la clause `GROUP BY` pour contourner le fait qu'une colonne doit faire partie de la clause `GROUP BY` ou d'une expression agrégée. Si cette règle n'est pas respectée, vous pouvez recevoir un message d'erreur du type suivant :

EXPRESSION_NOT_AGGREGATE : ligne 1:8 : « catégorie » doit être une expression agrégée ou apparaître dans la clause GROUP BY

Pour éviter d'avoir à ajouter des colonnes redondantes à la clause `GROUP BY`, vous pouvez utiliser la fonction ARBITRARY, comme dans l'exemple suivant.

```
SELECT country_id,
       arbitrary(country_name) AS country_name,
       COUNT(*) AS city_count
FROM world_cities
GROUP BY country_id
```

La fonction ARBITRARY renvoie une valeur arbitraire à partir du groupe. La fonction est utile lorsque vous savez que tous les enregistrements du groupe ont la même valeur pour une colonne, mais que cette valeur n'identifie pas le groupe.

Optimisation des requêtes Top N

La clause ORDER BY renvoie les résultats d'une requête dans un ordre trié. Athena utilise le tri distribué pour exécuter l'opération de tri en parallèle sur plusieurs nœuds.

Si vous n'avez pas strictement besoin que votre résultat soit trié, évitez d'ajouter une clause ORDER BY. Évitez également d'ajouter des clauses ORDER BY à des requêtes internes si elles ne sont pas strictement nécessaires. Dans de nombreux cas, le planificateur de requêtes peut supprimer le tri redondant, mais cela n'est pas garanti. Il existe une exception à cette règle si une requête interne effectue une opération Top N, telle que la recherche des valeurs N les plus récentes ou N les plus courantes.

Quand Athena voit ORDER BY en même temps que LIMIT, il comprend que vous exécutez une requête Top N et utilise des opérations dédiées en conséquence.

> **Note**
>
> Bien qu'Athena puisse également souvent détecter des fonctions de fenêtrage telles row_number qui utilisent N principal, nous recommandons la version plus simple qui utilise ORDER BY et LIMIT. Pour plus d'informations, consultez Optimisation des fonctions de fenêtrage.

Inclure uniquement les colonnes obligatoires

Si vous n'avez pas strictement besoin d'une colonne, ne l'incluez pas dans votre requête. Moins une requête doit traiter de données, plus elle sera exécutée rapidement. Cela réduit à la fois la quantité de mémoire requise et la quantité de données à envoyer entre les nœuds. Si vous utilisez un format de fichier en colonnes, la réduction du nombre de colonnes réduit également la quantité de données lues à partir d'Amazon S3.

Athena n'impose aucune limite spécifique quant au nombre de colonnes dans un résultat, mais la manière dont les requêtes sont exécutées limite la taille combinée possible des colonnes. La taille combinée des colonnes inclut leur nom et leur type.

Par exemple, l'erreur suivante est due à une relation qui dépasse la limite de taille d'un descripteur de relation :

ERREUR INTERNE GÉNÉRIQUE : io.airlift.bytecode. CompilationException

Pour contourner ce problème, réduisez le nombre de colonnes dans la requête ou créez des sous-requêtes et utilisez une commande `JOIN` qui récupère une plus petite quantité de données. Si vous avez des requêtes effectuant `SELECT *` dans la requête la plus externe, vous devez remplacer l'* par une liste contenant uniquement les colonnes dont vous avez besoin.

Optimisation des requêtes à l'aide d'approximations

Athena prend en charge les fonctions d'agrégation d'approximations pour compter les valeurs distinctes, les valeurs les plus fréquentes, les percentiles (y compris les médianes approximatives) et créer des histogrammes. Utilisez ces fonctions chaque fois que des valeurs exactes ne sont pas nécessaires.

Contrairement aux opérations `COUNT(DISTINCT col)`, approx_distinct utilise beaucoup moins de mémoire et s'exécute plus rapidement. De même, l'utilisation de numeric_histogram au lieu de histogram utilise des méthodes approximatives et donc moins de mémoire.

Optimisation de LIKE

Vous pouvez utiliser `LIKE` pour trouver des chaînes correspondantes, mais avec de longues chaînes, cela demande beaucoup de calcul. La fonction regexp_like est dans la plupart des cas une alternative plus rapide et offre également plus de flexibilité.

Vous pouvez souvent optimiser une recherche en ancrant la sous-chaîne que vous recherchez. Par exemple, si vous recherchez un préfixe, il est préférable d'utiliser « *`substr`*% » au lieu de « %*`substr`*% ». Ou, si vous utilisez `regexp_like` « ^ *`substr`* ».

Utiliser UNION ALL au lieu de UNION

`UNION ALL` et `UNION` sont deux manières de combiner les résultats de deux requêtes en un seul résultat. `UNION ALL` concatène les enregistrements de la première requête avec la seconde, et `UNION` fait de même, mais supprime également les doublons. `UNION` doit traiter tous les

enregistrements et trouver les doublons, ce qui demande beaucoup de mémoire et de calcul, mais `UNION ALL` est une opération relativement rapide. À moins que vous n'ayez besoin de dédupliquer des enregistrements, utilisez `UNION ALL` pour de meilleures performances.

Utiliser UNLOAD pour les grands ensembles de résultats

Lorsque les résultats d'une requête sont censés être volumineux (par exemple, des dizaines de milliers de lignes ou plus), utilisez UNLOAD pour exporter les résultats. Dans la plupart des cas, cela est plus rapide que l'exécution d'une requête normale, et l'utilisation de `UNLOAD` vous permet également de mieux contrôler le résultat.

Lorsque l'exécution d'une requête est terminée, Athena stocke le résultat sous la forme d'un seul fichier CSV non compressé sur Amazon S3. Cela prend plus de temps que `UNLOAD`, non seulement parce que le résultat n'est pas compressé, mais également parce que l'opération ne peut pas être parallélisée. En revanche, `UNLOAD` écrit les résultats directement à partir des composants master et utilise pleinement le parallélisme du cluster de calcul. En outre, vous pouvez configurer `UNLOAD` pour écrire les résultats au format compressé et dans d'autres formats de fichier tels que JSON et Parquet.

Pour plus d'informations, consultez UNLOAD.

Utiliser CTAS ou ETL Glue pour matérialiser les agrégations fréquemment utilisées

La « matérialisation » d'une requête est un moyen d'accélérer les performances des requêtes en stockant des résultats de requêtes complexes précalculés (par exemple, des agrégations et des jointures) pour les réutiliser dans les requêtes suivantes.

Si bon nombre de vos requêtes incluent les mêmes jointures et agrégations, vous pouvez matérialiser la sous-requête commune sous la forme d'une nouvelle table, puis exécuter des requêtes sur cette table. Vous pouvez créer la nouvelle table avec Création d'une table à partir des résultats des requêtes (CTAS) ou un outil ETL dédié tel que ETL Glue.

Supposons, par exemple, que vous disposiez d'un tableau de bord comprenant des widgets qui présentent différents aspects d'un jeu de données de commandes. Chaque widget possède sa propre requête, mais les requêtes partagent toutes les mêmes jointures et filtres. Une table des commandes est jointe à une table des articles, et un filtre permet de n'afficher que les trois derniers mois. Si vous identifiez les caractéristiques communes de ces requêtes, vous pouvez créer une nouvelle table que les widgets pourront utiliser. Cela réduit les doublons et améliore les performances. L'inconvénient est que vous devez maintenir la nouvelle table à jour.

Réutiliser les résultats des requêtes

Il est courant qu'une même requête soit exécutée plusieurs fois dans un court laps de temps. Cela peut se produire, par exemple, lorsque plusieurs personnes ouvrent le même tableau de bord de données. Lorsque vous exécutez une requête, vous pouvez demander à Athena de réutiliser les résultats précédemment calculés. Vous spécifiez l'âge maximal des résultats à réutiliser. Si la même requête a déjà été exécutée pendant cette période, Athena renvoie ces résultats au lieu de réexécuter la requête. Pour plus d'informations, consultez Réutilisation des résultats des requêtes ici dans le Guide de l'utilisateur Amazon Athena et Réduction des coûts et amélioration des performances des requêtes grâce à la réutilisation des résultats des requêtes Amazon Athena sur le blog AWS Big Data.

Techniques d'optimisation des données

Les performances dépendent non seulement des requêtes, mais aussi et surtout de la manière dont votre jeu de données est organisé, ainsi que du format de fichier et de la compression qu'il utilise.

Partitionner vos données

Le partitionnement divise votre table en plusieurs parties et conserve les données associées en fonction de propriétés telles que la date, le pays ou la région. Les clés de partition agissent comme des colonnes virtuelles. Vous définissez les clés de partition lors de la création de la table et vous les utilisez pour filtrer vos requêtes. Lorsque vous filtrez sur les colonnes de clés de partition, seules les données des partitions correspondantes sont lues. Par exemple, si votre jeu de données est partitionné par date et que votre requête comporte un filtre qui ne correspond qu'à la semaine dernière, seules les données de la dernière semaine sont lues. Pour plus d'informations sur le partitionnement, consultez Partitionnement de données dans Athena.

Sélectionner les clés de partition qui répondront à vos requêtes

Le partitionnement ayant un impact significatif sur les performances des requêtes, veillez à bien réfléchir à la manière dont vous partitionnez lorsque vous concevez votre jeu de données et vos tables. Un trop grand nombre de clés de partition peut entraîner la fragmentation des jeux de données contenant trop de fichiers et des fichiers trop petits. À l'inverse, le fait d'avoir trop peu de clés de partition ou de ne pas partitionner du tout, entraîne des requêtes qui analysent plus de données que nécessaire.

Éviter l'optimisation des requêtes rares

Une bonne stratégie consiste à optimiser pur les requêtes les plus courantes et à éviter d'optimiser les requêtes rares. Par exemple, si vos requêtes portent sur des périodes de plusieurs jours,

ne partitionnez pas par heure, même si certaines requêtes filtrent à ce niveau. Si vos données comportent une colonne d'horodatage précise, les rares requêtes filtrées par heure peuvent utiliser la colonne d'horodatage. Même si de rares cas analysent un peu plus de données que nécessaire, réduire les performances globales au profit de cas rares ne constitue généralement pas un bon compromis.

Pour réduire la quantité de données que les requêtes doivent analyser et améliorer ainsi les performances, utilisez un format de fichier en colonnes et triez les enregistrements. Au lieu de partitionner par heure, gardez les enregistrements triés par horodatage. Pour les requêtes portant sur des fenêtres temporelles plus courtes, le tri par horodatage est presque aussi efficace que le partitionnement par heure. En outre, le tri par horodatage ne nuit généralement pas aux performances des requêtes sur des fenêtres temporelles comptées en jours. Pour plus d'informations, consultez Utiliser les formats de fichier en colonnes.

Notez que les requêtes sur des tables contenant des dizaines de milliers de partitions sont plus performantes si toutes les clés de partition comportent des prédicats. C'est une autre raison de concevoir votre schéma de partitionnement pour les requêtes les plus courantes. Pour plus d'informations, consultez Interroger les partitions par égalité.

Utiliser la projection de partition

La projection de partition est une fonctionnalité d'Athena qui stocke les informations de partition non pas dans le AWS Glue Data Catalog, mais sous forme de règles dans les propriétés de la table dans. AWS Glue Lorsqu'Athena planifie une requête sur une table configurée avec une projection de partition, il lit les règles de projection de partition de la table. Athena calcule les partitions à lire en mémoire en fonction de la requête et des règles au lieu de rechercher des partitions dans le AWS Glue Data Catalog.

Outre la simplification de la gestion des partitions, la projection de partition peut améliorer les performances des jeux de données comportant un grand nombre de partitions. Lorsqu'une requête inclut des plages au lieu de valeurs spécifiques pour les clés de partition, la durée de la recherche des partitions correspondantes dans le catalogue est fonction du nombre de partitions. Avec la projection de partition, le filtre peut être calculé en mémoire sans passer par le catalogue, et cela peut être beaucoup plus rapide.

Dans certaines circonstances, la projection de partition peut entraîner une baisse des performances. Un exemple se produit lorsqu'une table est « fragmentée ». Une table fragmentée ne contient pas de données pour chaque permutation des valeurs de clé de partition décrite par la configuration de projection de partition. Avec une table fragmentée, l'ensemble de partitions calculé à partir de la

requête et la configuration de projection de partition sont tous répertoriés sur Amazon S3, même s'ils ne contiennent aucune donnée.

Lorsque vous utilisez la projection de partition, veillez à inclure des prédicats sur toutes les clés de partition. Restreignez la portée des valeurs possibles pour éviter des listes inutiles sur Amazon S3. Imaginez une clé de partition comportant une plage d'un million de valeurs et une requête sans filtre sur cette clé de partition. Pour exécuter la requête, Athena doit effectuer au moins un million d'opérations de liste Amazon S3. Les requêtes sont plus rapides lorsque vous interrogez des valeurs spécifiques, que vous utilisiez la projection de partition ou que vous stockiez les informations de partition dans le catalogue. Pour plus d'informations, consultez Interroger les partitions par égalité.

Lorsque vous configurez une table pour la projection de partition, assurez-vous que les plages que vous spécifiez sont raisonnables. Si une requête n'inclut pas de prédicat sur une clé de partition, toutes les valeurs de la plage correspondant à cette clé sont utilisées. Si votre jeu de données a été créé à une date précise, utilisez cette date comme point de départ pour toutes les plages de dates. Utiliser `NOW` comme fin des plages de dates. Évitez les plages numériques contenant un grand nombre de valeurs et envisagez plutôt d'utiliser le type injecté.

Pour plus d'informations sur la projection de partition, voir Projection de partition avec Amazon Athena.

Utiliser les index de partition

Les index de partition sont une fonctionnalité du AWS Glue Data Catalog qui améliore les performances de recherche de partitions pour les tables contenant un grand nombre de partitions.

La liste des partitions du catalogue est similaire à une table dans une base de données relationnelle. La table comporte des colonnes pour les clés de partition et une colonne supplémentaire pour l'emplacement de la partition. Lorsque vous interrogez une table partitionnée, les emplacements des partitions sont recherchés en analysant cette table.

Tout comme pour les bases de données relationnelles, vous pouvez améliorer les performances des requêtes en ajoutant des index. Vous pouvez ajouter plusieurs index pour prendre en charge différents modèles de requêtes. L'index de AWS Glue Data Catalog partition prend en charge à la fois les opérateurs d'égalité et de comparaison tels que >>=, et < combinés avec l'`AND`opérateur. Pour plus d'informations, consultez les sections Utilisation des index de partition AWS Glue dans le guide du AWS Glue développeur et amélioration des performances des requêtes Amazon Athena à AWS Glue Data Catalog l'aide d'index de partition dans AWS le blog Big Data.

Toujours utiliser STRING comme type pour les clés de partition

Lorsque vous interrogez les clés de partition, n'oubliez pas qu'Athena exige que les clés de partition soient de type `STRING` afin de pousser vers le bas le filtrage des partitions dans AWS Glue. Si le nombre de partitions n'est pas faible, l'utilisation d'autres types peut entraîner une baisse des performances. Si les valeurs de vos clés de partition sont de type date ou numérique, convertissez-les dans le type approprié dans votre requête.

Supprimer les partitions anciennes et vides

Si vous supprimez des données d'une partition sur Amazon S3 (par exemple, en utilisant le cycle de vie d'Amazon S3), vous devez également supprimer l'entrée de partition du AWS Glue Data Catalog. Lors de la planification des requêtes, toute partition correspondant à la requête est répertoriée sur Amazon S3. Si vous avez de nombreuses partitions vides, la surcharge liée à la liste de ces partitions peut être préjudiciable.

De même, si vous avez plusieurs milliers de partitions, pensez à supprimer les métadonnées de partition pour les anciennes données qui ne sont plus pertinentes. Par exemple, si les requêtes ne portent jamais sur des données datant de plus d'un an, vous pouvez régulièrement supprimer les métadonnées des partitions les plus anciennes. Si le nombre de partitions atteint des dizaines de milliers, la suppression des partitions inutilisées peut accélérer les requêtes qui n'incluent pas de prédicats sur toutes les clés de partition. Pour plus d'informations sur l'inclusion de prédicats sur toutes les clés de partition dans vos requêtes, consultez Interroger les partitions par égalité.

Interroger les partitions par égalité

Les requêtes qui incluent des prédicats d'égalité sur toutes les clés de partition s'exécutent plus rapidement, car les métadonnées de partition peuvent être chargées directement. Évitez les requêtes dans lesquelles une ou plusieurs clés de partition n'ont pas de prédicat ou dans lesquelles le prédicat sélectionne une plage de valeurs. Pour de telles requêtes, la liste de toutes les partitions doit être filtrée pour trouver les valeurs correspondantes. Pour la plupart des tables, la surcharge est minime, mais pour les tables comportant des dizaines de milliers de partitions ou plus, elle peut devenir importante.

S'il n'est pas possible de réécrire vos requêtes pour filtrer les partitions par égalité, vous pouvez essayer la projection de partition. Pour plus d’informations, consultez Utiliser la projection de partition.

Éviter d'utiliser MSCK REPAIR TABLE pour la maintenance des partitions

Étant donné que l'exécution de `MSCK REPAIR TABLE` peut prendre du temps, qu'elle ajoute uniquement de nouvelles partitions et qu'elle ne supprime pas les anciennes partitions, ce n'est pas une méthode efficace pour gérer les partitions (voir Considérations et restrictions).

Il est préférable de gérer les partitions manuellement à l'aide d'API AWS Glue Data Catalog, ALTER TABLE ADD PARTITION ou de Crawlers AWS Glue. Vous pouvez également utiliser la projection de partition, qui élimine complètement le besoin de gérer les partitions. Pour plus d'informations, consultez Projection de partition avec Amazon Athena.

Vérifiez que vos requêtes sont compatibles avec le schéma de partitionnement

Vous pouvez vérifier à l'avance les partitions qu'une requête analysera à l'aide de l'instruction `EXPLAIN`. Préfixez votre requête avec le mot-clé `EXPLAIN`, puis recherchez le fragment source (par exemple,`Fragment 2 [SOURCE]`) pour chaque table en bas de du résultat `EXPLAIN`. Recherchez les affectations dont la partie droite est définie comme clé de partition. La ligne ci-dessous inclut une liste de toutes les valeurs de cette clé de partition qui seront analysées lors de l'exécution de la requête.

Supposons, par exemple, que vous ayez une requête sur une table contenant une clé de partition `dt` et que vous la préfixiez avec `EXPLAIN`. Si les valeurs de la requête sont des dates et qu'un filtre sélectionne une plage de trois jours, le résultat `EXPLAIN` peut ressembler à ceci :

```
dt := dt:string:PARTITION_KEY
    :: [[2023-06-11], [2023-06-12], [2023-06-13]]
```

Le résultat `EXPLAIN` indique que le planificateur a trouvé trois valeurs pour cette clé de partition correspondant à la requête. Il vous indique également quelles sont ces valeurs. Pour plus d'informations sur l'utilisation de `EXPLAIN` et de Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena, consultez Explication des résultats de l'instruction EXPLAIN d'Athena.

Utiliser les formats de fichier en colonnes

Les formats de fichiers en colonnes tels que Parquet et ORC sont conçus pour les charges de travail d'analyse distribuées. Ils organisent les données par colonne plutôt que par ligne. L'organisation des données sous forme de colonnes présente les avantages suivants :

- Seules les colonnes nécessaires à la requête sont chargées
- La quantité globale de données à charger est réduite

- Les valeurs des colonnes sont stockées ensemble, de sorte que les données peuvent être compressées efficacement
- Les fichiers peuvent contenir des métadonnées qui permettent au moteur d'éviter de charger des données inutiles

À titre d'exemple de la manière dont les métadonnées des fichiers peuvent être utilisées, les métadonnées des fichiers peuvent contenir des informations sur les valeurs minimales et maximales d'une page de données. Si les valeurs demandées ne se situent pas dans la plage indiquée dans les métadonnées, la page peut être ignorée.

L'un des moyens d'utiliser ces métadonnées pour améliorer les performances consiste à s'assurer que les données contenues dans les fichiers sont triées. Supposons, par exemple, que vous ayez des requêtes qui recherchent des enregistrements dont l'entrée `created_at` se trouve dans un court laps de temps. Si vos données sont triées par colonne `created_at`, Athena peut utiliser les valeurs minimale et maximale des métadonnées des fichiers pour ignorer les parties inutiles des fichiers de données.

Lorsque vous utilisez des formats de fichier en colonnes, assurez-vous que vos fichiers ne sont pas trop petits. Comme indiqué dans Éviter d'avoir trop de fichiers, les jeux de données contenant de nombreux petits fichiers entraînent des problèmes de performances. Cela est particulièrement vrai pour les formats de fichiers en colonnes. Pour les petits fichiers, la surcharge du format de fichier en colonnes l'emporte sur les avantages.

Notez que Parquet et ORC sont organisés en interne par groupes de lignes (Parquet) et de bandes (ORC). La taille par défaut pour les groupes de lignes est de 128 Mo et pour les bandes, de 64 Mo. Si vous avez de nombreuses colonnes, vous pouvez augmenter la taille des groupes de lignes et des bandes pour de meilleures performances. Il n'est pas recommandé de réduire la taille des groupes de lignes ou des bandes à une valeur inférieure à leurs valeurs par défaut.

Pour convertir d'autres formats de données en Parquet ou ORC, vous pouvez utiliser AWS Glue ETL ou Athena. Pour plus d'informations sur l'utilisation d'Athena pour ETL, consultez Utilisation de CTAS et INSERT INTO pour ETL et l'analyse des données.

Compresser les données

Athena prend en charge un large éventail de formats de compression. L'interrogation de données compressées est plus rapide et moins coûteuse, car vous payez pour le nombre d'octets analysés avant la décompression.

Le format gzip fournit de bons taux de compression et est largement compatible avec d'autres outils et services. Le format zstd (Zstandard) est un format de compression plus récent offrant un bon équilibre entre performances et taux de compression.

Lorsque vous compressez des fichiers texte tels que des données JSON et CSV, essayez de trouver un équilibre entre le nombre de fichiers et leur taille. La plupart des formats de compression nécessitent que le lecteur lise les fichiers dès le début. Cela signifie que les fichiers texte compressés ne peuvent généralement pas être traités en parallèle. Les fichiers non compressés volumineux sont souvent répartis entre les outils de traitement afin d'obtenir un parallélisme accru lors du traitement des requêtes, mais cela n'est pas possible avec la plupart des formats de compression.

Comme indiqué dans Éviter d'avoir trop de fichiers, il est préférable de n'avoir ni trop ni trop peu de fichiers. Le nombre de fichiers étant la limite du nombre de travailleurs autorisés à traiter la requête, cette règle est particulièrement vraie pour les fichiers compressés.

Pour plus d'informations sur l'utilisation de la compression dans Athena, consultez Prise en charge de la compression Athena.

Utiliser la compartimentation pour les recherches sur des clés à cardinalité élevée

La compartimentation est une technique qui permet de répartir les enregistrements dans des fichiers séparés en fonction de la valeur de l'une des colonnes. Cela garantit que tous les enregistrements ayant la même valeur figurent dans le même fichier. La compartimentation est utile lorsque vous avez une clé à cardinalité élevée et que bon nombre de vos requêtes recherchent des valeurs spécifiques de cette clé.

Supposons, par exemple, que vous interrogiez un ensemble d'enregistrements pour un utilisateur spécifique. Si les données sont compartimentées par nom d'utilisateur, Athena sait à l'avance quels fichiers contiennent des enregistrements pour un identifiant spécifique et quels fichiers n'en contiennent pas. Cela permet à Athena de lire uniquement les fichiers pouvant contenir l'identifiant, ce qui réduit considérablement le volume de données lues. Cela réduit également le temps de calcul qui serait autrement nécessaire pour rechercher l'identifiant spécifique dans les données.

Inconvénients de la compartimentation

La compartimentation est moins utile lorsque les requêtes recherchent fréquemment plusieurs valeurs dans la colonne dans laquelle les données sont compartimentées. Plus le nombre de valeurs demandées est élevé, plus il est probable que tous les fichiers ou la plupart d'entre eux devront être lus. Par exemple, si vous avez trois compartiments et qu'une requête recherche trois valeurs

différentes, il est possible que tous les fichiers doivent être lus. La compartimentation fonctionne mieux lorsque les requêtes recherchent des valeurs uniques.

Pour plus d'informations, consultez Partitionnement et compartimentation dans Athena.

Éviter d'avoir trop de fichiers

Les jeux de données composés de nombreux petits fichiers se traduisent par de faibles performances globales pour les requêtes. Lorsqu'Athena planifie une requête, il répertorie tous les emplacements des partitions, ce qui prend du temps. La gestion et la demande de chaque fichier entraînent également une surcharge de calcul. Par conséquent, le chargement d'un seul fichier plus volumineux à partir d'Amazon S3 est plus rapide que le chargement des mêmes enregistrements à partir de nombreux fichiers plus petits.

Dans des cas extrêmes, vous pouvez rencontrer des limites de service Amazon S3. Amazon S3 prend en charge jusqu'à 5 500 demandes par seconde adressées à une seule partition d'index. Au départ, un compartiment est traité comme une seule partition d'index, mais à mesure que la charge de demandes augmente, il peut être divisé en plusieurs partitions d'index.

Amazon S3 examine les modèles de demandes et les fractionne en fonction de préfixes clés. Si votre jeu de données se compose de plusieurs milliers de fichiers, les demandes provenant d'Athena peuvent dépasser le quota de demandes. Même avec moins de fichiers, le quota peut être dépassé si plusieurs requêtes simultanées sont effectuées sur le même jeu de données. Les autres applications qui accèdent aux mêmes fichiers peuvent contribuer au nombre total de demandes.

Lorsque le taux de demandes `limit` est dépassé, Amazon S3 renvoie l'erreur suivante. Cette erreur est incluse dans les informations d'état de la requête dans Athena.

SlowDown: Veuillez réduire votre taux de demandes

Pour résoudre le problème, commencez par déterminer si l'erreur est due à une seule requête ou à plusieurs requêtes qui lisent les mêmes fichiers. Dans ce dernier cas, coordonnez l'exécution des requêtes afin qu'elles ne s'exécutent pas en même temps. Pour ce faire, ajoutez un mécanisme de mise en file d'attente ou réessayez dans votre application.

Si l'exécution d'une seule requête déclenche l'erreur, essayez de combiner des fichiers de données ou de modifier la requête pour lire moins de fichiers. Il est préférable de combiner les petits fichiers avant leur écriture. Pour ce faire, envisagez les techniques suivantes :

- Modifiez le processus d'écriture des fichiers pour écrire des fichiers plus volumineux. Par exemple, vous pouvez mettre les enregistrements en mémoire tampon plus longtemps avant qu'ils ne soient écrits.
- Placez les fichiers dans un emplacement sur Amazon S3 et utilisez un outil tel que ETL Glue pour les combiner en fichiers plus volumineux. Déplacez ensuite les fichiers les plus volumineux vers l'emplacement indiqué dans la table. Pour plus d'informations, consultez les sections Lecture de fichiers d'entrée en groupes plus importants dans le guide du AWS Glue développeur et Comment configurer une tâche AWS Glue ETL pour générer des fichiers plus volumineux ? dans le centre de connaissances AWS Re:post.
- Réduisez le nombre de clés de partition. Lorsque vous avez trop de clés de partition, chaque partition peut ne contenir que quelques enregistrements, ce qui entraîne un nombre excessif de petits fichiers. Pour plus d'informations sur le choix des partitions à créer, consultez Sélectionner les clés de partition qui répondront à vos requêtes.

Éviter les hiérarchies de stockage supplémentaires au-delà de la partition

Pour éviter de surcharger la planification des requêtes, stockez les fichiers dans une structure plate à l'emplacement de chaque partition. N'utilisez aucune hiérarchie de répertoire supplémentaire.

Lorsqu'Athena planifie une requête, il répertorie tous les fichiers de toutes les partitions correspondant à la requête. Bien qu'Amazon S3 ne dispose pas de répertoires en soi, la convention consiste à interpréter la barre oblique / comme séparateur de répertoires. Lorsqu'Athena répertorie les emplacements des partitions, il répertorie de manière récursive tous les répertoires qu'il trouve. Lorsque les fichiers d'une partition sont organisés en hiérarchie, plusieurs séries de listes ont lieu.

Lorsque tous les fichiers se trouvent directement à l'emplacement de la partition, la plupart du temps, une seule opération de liste doit être effectuée. Cependant, plusieurs opérations de liste séquentielles sont nécessaires si vous avez plus de 1 000 fichiers dans une partition, car Amazon S3 ne renvoie que 1 000 objets par opération de liste. La présence de plus de 1 000 fichiers dans une partition peut également créer d'autres problèmes de performances plus graves. Pour plus d'informations, consultez Éviter d'avoir trop de fichiers.

Utiliser SymlinkTextInputFormat uniquement lorsque cela est nécessaire

L'utilisation de la technique SymlinkTextInputFormat peut être un moyen de contourner les situations dans lesquelles les fichiers d'une table ne sont pas bien organisés en partitions. Par exemple, les liens symboliques peuvent être utiles lorsque tous les fichiers se trouvent dans le même préfixe ou lorsque des fichiers avec des schémas différents se trouvent au même emplacement.

Cependant, l'utilisation de liens symboliques ajoute des niveaux d'indirection à l'exécution de la requête. Ces niveaux d'indirection ont un impact sur les performances globales. Les fichiers de liens symboliques doivent être lus et les emplacements qu'ils définissent doivent être répertoriés. Cela ajoute plusieurs allers-retours vers Amazon S3, ce qui n'est pas nécessaire pour les tables Hive habituelles. En conclusion, vous ne devez utiliser `SymlinkTextInputFormat` que lorsque de meilleures options, telles que la réorganisation des fichiers, ne sont pas disponibles.

Ressources supplémentaires

Pour plus d'informations sur le réglage des performances dans Athena, consultez les ressources suivantes :

- Lisez le billet de blog consacré au AWS Big Data : les 10 meilleurs conseils pour optimiser les performances d'Amazon Athena
- Pour un article sur l'utilisation de la poussée vers le bas de prédicats pour améliorer les performances des requêtes fédérées, consultez Améliorer les requêtes fédérées avec la poussée vers le bas de prédicats dans Amazon Athena sur le blog AWS Big Data.
- Pour consulter un article sur les optimisations des performances du moteur de requêtes Athena, voir Exécuter des requêtes 3 fois plus vite avec jusqu'à 70 % d'économies sur le dernier moteur Amazon Athena sur le blog Big Data.AWS
- Lisez d'autres articles d'Athena sur le blog consacré aux AWS mégadonnées
- Poser une question sur AWS re:Post en utilisant l'étiquette Amazon Athena
- Consultez les sujets relatifs à Athena dans le centre de connaissances AWS
- Contact AWS Support (dans le AWS Management Console, cliquez sur Support, Centre de support)

Prévention de la limitation Amazon S3

La limitation est le processus qui consiste à limiter le taux d'utilisation d'un service, d'une application ou d'un système. Dans AWS, vous pouvez utiliser la régulation pour empêcher la surutilisation du service Amazon S3 et augmenter la disponibilité et la réactivité d'Amazon S3 pour tous les utilisateurs. Cependant, étant donné que la limitation restreint la vitesse à laquelle les données peuvent être transférées vers ou depuis Amazon S3, il est important d'en tenir compte pour éviter que vos interactions ne soient limitées.

Réduire la limitation au niveau du service

Pour éviter la limitation d'Amazon S3 au niveau du service, vous pouvez surveiller votre utilisation et ajuster vos Service Quotas, ou utiliser certaines techniques telles que le partitionnement. Voici certaines des conditions qui peuvent entraîner une limitation :

- Dépassement des limites de demandes d'API de votre compte : Amazon S3 a des limites de demandes d'API par défaut qui sont basées sur le type de compte et son utilisation. Si vous dépassez le nombre maximum de demandes par seconde pour un seul objet, vos demandes peuvent être limitées afin d'éviter une surcharge du service Amazon S3.
- Partitionnement insuffisant des données : si vous ne partitionnez pas correctement vos données et que vous transférez une grande quantité de données, Amazon S3 peut limiter vos demandes. Pour plus d'informations sur le partitionnement, consultez la section Utiliser le partitionnement plus haut dans ce document.
- Grand nombre de petits objets : si possible, évitez d'avoir un grand nombre de petits fichiers. Amazon S3 a une limite de 5 500 demandes GET par seconde et par préfixe partitionné, et vos requêtes Athena partagent cette même limite. Si vous analysez des millions de petits objets en une seule requête, votre requête peut être facilement limitée par Amazon S3.

Pour éviter une analyse excessive, vous pouvez utiliser l' AWS Glue ETL pour compacter régulièrement vos fichiers, ou vous pouvez partitionner la table et ajouter des filtres de clé de partition. Pour plus d'informations, veuillez consulter les ressources suivantes.

- Comment configurer une tâche AWS Glue ETL pour générer des fichiers plus volumineux ? (Centre de AWS connaissances)
- Lecture de fichiers d'entrée en groupes plus importants (Guide AWS Glue du développeur)

Optimisation de vos tables

Il est important de structurer vos données si vous rencontrez des problèmes de limitation. Bien qu'Amazon S3 puisse gérer de grandes quantités de données, une limitation se produit parfois en raison de la structure des données.

Les sections suivantes proposent des suggestions sur la manière de structurer vos données dans Amazon S3 afin d'éviter les problèmes de limitation.

Utiliser le partitionnement

Vous pouvez utiliser le partitionnement pour réduire la limitation en limitant la quantité de données auxquelles il est nécessaire d'accéder à un moment donné. En partitionnant les données sur des colonnes spécifiques, vous pouvez répartir les demandes de manière uniforme sur plusieurs objets et réduire le nombre de demandes pour un seul objet. La réduction de la quantité de données devant être analysées permet d'améliorer les performances des requêtes et de réduire les coûts.

Vous pouvez définir des partitions, qui agissent comme des colonnes virtuelles, lorsque vous créez une table. Pour créer une table avec des partitions dans une instruction `CREATE TABLE`, vous utilisez la clause `PARTITIONED BY (`*`column_name data_type`*`)` pour définir les clés permettant de partitionner vos données.

Pour restreindre les partitions analysées par une requête, vous pouvez les spécifier sous forme de prédicats dans une clause `WHERE` de la requête. Les colonnes fréquemment utilisées comme filtres se prêtent donc parfaitement au partitionnement. Il est d'usage de partitionner les données en fonction de l'heure, ce qui peut conduire à un schéma de partitionnement à plusieurs niveaux.

Notez que le partitionnement a également un coût. Lorsque vous augmentez le nombre de partitions dans votre table, le temps nécessaire pour récupérer et traiter les métadonnées des partitions augmente également. Ainsi, le surpartitionnement peut supprimer les avantages que vous obtenez en partitionnant de manière plus judicieuse. Si vos données sont fortement asymétriques par rapport à une valeur de partition et que la plupart des requêtes utilisent cette valeur, vous risquez d'avoir à supporter des frais généraux supplémentaires.

Pour plus d'informations sur le partitionnement dans Athena, consultez Qu'est-ce que le partitionnement ?.

Compartimenter vos données

Une autre façon de partitionner vos données consiste à les compartimenter dans une seule partition. La compartimentation vous permet de spécifier une ou plusieurs colonnes contenant des lignes que vous souhaitez regrouper. Ensuite, vous placez ces lignes dans plusieurs compartiments. De cette manière, vous interrogez uniquement le compartiment qui doit être lu, ce qui réduit le nombre de lignes de données devant être analysées.

Lorsque vous sélectionnez une colonne à utiliser pour la compartimentation, sélectionnez la colonne présentant une cardinalité élevée (c'est-à-dire comportant de nombreuses valeurs distinctes), distribuée uniformément et fréquemment utilisée pour filtrer les données. Une clé primaire, telle qu'une colonne ID, est un exemple de colonne appropriée à utiliser pour la compartimentation.

Pour plus d'informations sur la compartimentation dans Athena, consultez Qu'est-ce que la compartimentation ?.

Utiliser des index AWS Glue de partition

Vous pouvez utiliser les index de AWS Glue partition pour organiser les données dans une table en fonction des valeurs d'une ou de plusieurs partitions. AWS Glue les index de partition peuvent réduire le nombre de transferts de données, la quantité de données traitées et le temps de traitement des requêtes.

Un index de AWS Glue partition est un fichier de métadonnées qui contient des informations sur les partitions de la table, notamment les clés de partition et leurs valeurs. L'index de partition est stocké dans un compartiment Amazon S3 et est mis à jour automatiquement au AWS Glue fur et à mesure que de nouvelles partitions sont ajoutées à la table.

Lorsqu'un index de AWS Glue partition est présent, les requêtes tentent de récupérer un sous-ensemble des partitions au lieu de charger toutes les partitions de la table. Les requêtes s'exécutent uniquement sur le sous-jeu de données correspondant à la requête.

Lorsque vous créez une table dans AWS Glue, vous pouvez créer un index de partition sur n'importe quelle combinaison de clés de partition définie sur la table. Après avoir créé un ou plusieurs index de partition sur une table, vous devez y ajouter une propriété permettant le filtrage des partitions. Ensuite, vous pouvez interroger la table à partir d'Athena.

Pour plus d'informations sur la création d'index de partition dans AWS Glue, consultez la section Utilisation des index de partition AWS Glue dans le Guide du AWS Glue développeur. Pour plus d'informations sur l'ajout d'une propriété de table pour activer le filtrage des partitions, consultez AWS Glue indexation et filtrage des partitions.

Utiliser la compression des données et le fractionnement des fichiers

La compression des données peut accélérer considérablement les requêtes si les fichiers ont une taille optimale ou s'ils peuvent être fractionnés en groupes logiques. En général, des taux de compression élevés nécessitent plus de cycles d'UC pour compresser et décompresser les données. Pour Athena, nous vous recommandons d'utiliser Apache Parquet ou Apache ORC, qui compressent les données par défaut. Pour plus d'informations sur la compression des données dans Athena, veuillez consulter Prise en charge de la compression Athena.

Le fractionnement de fichiers augmente le parallélisme en permettant à Athena de répartir la tâche de lecture d'un seul fichier entre plusieurs lecteurs. Si un fichier unique n'est pas fractionnable, seul un

lecteur peut lire le fichier tandis que les autres lecteurs sont inactifs. Apache Parquet et Apache ORC prennent également en charge les fichiers fractionnables.

Utiliser les magasins de données en colonnes optimisés

Les performances des requêtes Athena s'améliorent de manière significative si vous convertissez vos données dans un format en colonnes. Lorsque vous générez des fichiers en colonnes, l'une des techniques d'optimisation à prendre en compte consiste à ordonner les données en fonction de la clé de partition.

Apache Parquet et Apache ORC sont des magasins de données en colonnes open source couramment utilisés. Pour plus d'informations sur la conversion d'une source de données Amazon S3 existante vers l'un de ces formats, consultez Conversion en formats de colonne.

Utiliser une taille supérieure de bloc Parquet ou de bande ORC

Parquet et ORC disposent de paramètres de stockage de données que vous pouvez régler pour les optimiser. Dans Parquet, vous pouvez optimiser la taille des blocs. Dans ORC, vous pouvez optimiser la taille des bandes. Plus le bloc ou la bande est grand(e), plus vous pouvez stocker de lignes dans chaque bloc ou bande. Par défaut, la taille de bloc Parquet est de 128 Mo et la taille de bande ORC est de 64 Mo.

Si une bande ORC est inférieure à 8 Mo (valeur par défaut de `hive.orc.max_buffer_size`), Athena lit l'intégralité de la bande ORC. C'est le compromis qu'Athena fait entre la sélectivité des colonnes et les opérations d'entrée/sortie par seconde pour les bandes plus petites.

Si vous avez des tables comportant un très grand nombre de colonnes, une petite taille de bloc ou de bande peut entraîner l'analyse d'un plus grand nombre de données que nécessaire. Dans ces cas, une taille de bloc plus grande peut s'avérer plus efficace.

Utiliser ORC pour les types complexes

Actuellement, lorsque vous interrogez les colonnes stockées dans Parquet qui ont des types de données complexes (par exemple `array`, `map` ou `struct`), Athena lit une ligne de données entière au lieu de lire sélectivement les colonnes spécifiées. Il s'agit d'un problème connu dans Athena. Pour contourner le problème, pensez à utiliser ORC.

Choix d'un algorithme de compression

Un autre paramètre que vous pouvez configurer est l'algorithme de compression des blocs de données. Pour plus d'informations sur les algorithmes de compression pris en charge pour Parquet et ORC dans Athena, consultez Prise en charge de la compression Athena.

Pour plus d'informations sur l'optimisation des formats de stockage en colonnes dans Athena, consultez la section « Optimiser la génération de banques de données en colonnes » dans AWS le billet de blog Big Data Les 10 meilleurs conseils d'optimisation des performances pour Amazon Athena.

Utiliser les tables Iceberg

Apache Iceberg est un format de table ouvert pour les jeux de données analytiques très volumineux conçu pour une utilisation optimisée sur Amazon S3. Vous pouvez utiliser les tables Iceberg pour permettre de réduire la limitation dans Amazon S3.

Les tables Iceberg vous offrent les avantages suivants :

- Vous pouvez partitionner les tables Iceberg sur une ou plusieurs colonnes. Cela permet d'optimiser l'accès aux données et de réduire la quantité de données devant être analysées par les requêtes.
- Comme le mode de stockage d'objets Iceberg optimise les tables Iceberg pour qu'elles fonctionnent avec Amazon S3, il peut traiter des volumes de données importants et de lourdes charges de travail liées aux requêtes.
- Les tables Iceberg en mode de stockage d'objets sont évolutives, tolérantes aux pannes et durables, ce qui peut contribuer à réduire la limitation.
- La prise en charge des transactions ACID signifie que plusieurs utilisateurs peuvent ajouter et supprimer des objets Amazon S3 de manière atomique.

Pour plus d'informations sur Apache Iceberg, consultez Apache Iceberg. Pour plus d'informations sur l'utilisation de tables Apache Iceberg dans Athena, consultez Utilisation des tables Iceberg.

Optimisation des requêtes

Utilisez les suggestions de cette section pour optimiser vos requêtes SQL dans Athena.

Utiliser LIMIT avec la clause ORDER BY

La clause `ORDER BY` renvoie les données dans un ordre trié. Cela oblige Athena à envoyer toutes les lignes de données à un seul composant master, puis à trier les lignes. Ce type de requête peut s'exécuter pendant une longue période, voire échouer.

Pour plus d'efficacité dans vos requêtes, examinez les valeurs *N* supérieures ou inférieures, puis utilisez également une clause `LIMIT`. Cela réduit considérablement le coût du tri en poussant le tri et la limitation vers des composants master individuels plutôt qu'à un seul composant master.

Optimiser les clauses JOIN

Lorsque vous joignez deux tables, Athena répartit la table de droite sur les composants master, puis diffuse la table de gauche pour effectuer la jointure.

Pour cette raison, spécifiez la table la plus grande du côté gauche de la jointure et la plus petite du côté droit de la jointure. De cette manière, Athena utilise moins de mémoire et exécute la requête avec une latence plus faible.

Notez également les points suivants :

- Lorsque vous utilisez plusieurs commandes `JOIN`, spécifiez les tables de la plus grande à la plus petite.
- Évitez les jointures croisées, sauf si elles sont requises par la requête.

Optimiser les clauses GROUP BY

L'opérateur `GROUP BY` répartit les lignes en fonction des colonnes `GROUP BY` sur les composants master. Ces colonnes sont référencées en mémoire et les valeurs sont comparées au fur et à mesure que les lignes sont ingérées. Les valeurs sont agrégées lorsque la colonne `GROUP BY` correspond. Compte tenu du fonctionnement de ce processus, il est conseillé d'ordonner les colonnes de la cardinalité la plus élevée à la plus faible.

Utiliser des nombres au lieu des chaînes

Comme les nombres nécessitent moins de mémoire et sont plus rapides à traiter que les chaînes, utilisez des nombres plutôt que des chaînes lorsque cela est possible.

Limiter le nombre de colonnes

Pour réduire la quantité totale de mémoire requise pour stocker vos données, limitez le nombre de colonnes spécifié dans votre instruction `SELECT`.

Utiliser des expressions régulières au lieu de LIKE

Les requêtes qui incluent des clauses, par exemple `LIKE '%string%'` sur de grandes chaînes, peuvent être très gourmandes en calculs. Lorsque vous filtrez plusieurs valeurs sur une colonne de chaîne, utilisez plutôt la fonction regexp_like() et une expression régulière. Cela est particulièrement utile lorsque vous comparez une longue liste de valeurs.

Utiliser la clause LIMIT

Au lieu de sélectionner toutes les colonnes lorsque vous exécutez une requête, utilisez la clause `LIMIT` pour renvoyer uniquement celles dont vous avez besoin. Cela réduit la taille du jeu de données traité via le pipeline d'exécution des requêtes. Les clauses `LIMIT` sont plus utiles lorsque vous interrogez des tables comportant un grand nombre de colonnes basées sur des chaînes. Les clauses `LIMIT` sont également utiles lorsque vous effectuez plusieurs jointures ou agrégations sur une requête.

Ressources supplémentaires

Schémas de conception des bonnes pratiques : optimisation des performances Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service

Réglage de performances dans Athena

Prise en charge de la compression Athena

Rubriques

Athena prend en charge divers formats de compression pour la lecture et l'écriture de données, y compris la lecture d'une table qui utilise plusieurs formats de compression. Par exemple, Athena peut lire avec succès les données d'une table qui utilise le format de fichier Parquet lorsque certains fichiers Parquet sont compressés avec Snappy et d'autres fichiers Parquet sont compressés avec GZIP. Le même principe s'applique aux formats de stockage ORC, fichier texte et JSON.

Athena prend en charge les formats de compression suivants :

- BZPI2 – Format qui utilise l'algorithme Burrows-Wheeler.
- DEFLATE – Algorithme de compression basé sur LZSS et sur le codage Huffman. Deflate n'est pertinent que pour le format de fichier Avro.

- GZIP – Algorithme de compression basé sur Deflate. Pour les tables Hive dans les versions 2 et 3 du moteur Athena, et les tables Iceberg dans la version 2 du moteur Athena, GZIP est le format de compression d'écriture par défaut des fichiers aux formats de stockage de fichier texte et Parquet. Les fichiers au format `tar.gz` ne sont pas pris en charge.
- LZ4 – Ce membre de la famille Lempel-Ziv 77 (LZ7) se concentre également sur la vitesse de compression et de décompression plutôt que sur la compression maximale des données. LZ4 possède les formats de cadrage suivants :
  - LZ4 Raw/Unframed (Brut/Non cadré) – Une implémentation standard, sans cadre, du format de compression de blocs LZ4. Pour plus d'informations, consultez la description du format de bloc LZ4 sur GitHub.
  - LZ4 cadré – L'implémentation habituelle tramée de LZ4. Pour plus d'informations, consultez la description du format de trame LZ4 sur GitHub.
  - Compatible Hadoop LZ4 – L'implémentation Apache Hadoop de LZ4. Cette implémentation intègre la compression LZ4 à la classe BlockCompressorStream.java.
- LZO – Format utilisant l'algorithme Lempel–Ziv–Oberhumer, qui se concentre sur une vitesse de compression et de décompression élevée plutôt que sur la compression maximale des données. LZO a deux implémentations :
  - LZO standard – Pour en savoir plus sur, consultez le résumé LZO sur le site web d'Oberhumer.
  - Compatible avec Hadoop LZO — Cette implémentation intègre l'algorithme LZO à la classe .java. BlockCompressorStream
- SNAPPY – Algorithme de compression faisant partie de la famille Lempel-Ziv 77 (LZ7). Snappy met l'accent sur une vitesse de compression et de décompression élevée plutôt que sur la compression maximale des données.
- ZLIB – Basé sur Deflate, ZLIB est le format de compression en écriture par défaut pour les fichiers au format de stockage de données ORC. Pour plus d'informations, consultez la page zlib sur GitHub.
- ZSTD – L'algorithme de compression de données en temps réel Zstandard est un algorithme de compression rapide qui fournit des taux de compression élevés. La bibliothèque Zstandard (ZSTD) est fournie sous forme de logiciel open source utilisant une licence BSD. ZSTD est la compression par défaut pour les tables Iceberg. Lors de l'écriture de données compressées ZSTD, Athena utilise par défaut le niveau 3 de compression ZSTD. Pour plus d'informations sur l'utilisation des niveaux de compression ZSTD dans Athena, consultez Utilisation des niveaux de compression ZSTD dans Athena.

Spécification des formats de compression

Lorsque vous écrivez des instructions CREATE TABLE ou CTAS, vous pouvez spécifier des propriétés de compression qui précisent le type de compression à utiliser lorsque Athena écrit dans ces tables.

- Pour CTAS, consultez Propriétés de la table CTAS. Pour obtenir des exemples, consultez Exemples de requêtes CTAS.
- Pour CREATE TABLE, consultez ALTER TABLE SET TBLPROPERTIES pour obtenir la liste des propriétés de la table de compression.

Spécification de l'absence de compression

Les instructions CREATE TABLE prennent en charge l'écriture de fichiers non compressés. Pour écrire des fichiers non compressés, utilisez la syntaxe suivante :

- CREATE TABLE (fichier texte ou JSON) — Dans `TBLPROPERTIES`, spécifiez `write.compression = NONE`.
- CREATE TABLE (Parquet) — Dans `TBLPROPERTIES`, spécifiez `parquet.compression = UNCOMPRESSED`.
- CREATE TABLE (ORC) — Dans `TBLPROPERTIES`, spécifiez `orc.compress = NONE`.

Remarques et ressources

- Actuellement, les extensions de fichier en majuscules telles que `.GZ` ou `.BZIP2` ne sont pas reconnues par Athena. Évitez d'utiliser des jeux de données avec des extensions de fichier en majuscules ou renommez les extensions des fichiers de données en minuscules.
- Pour des données aux formats CSV, TSV et JSON, Athena détermine le type de compression à partir de l'extension de fichier. Si aucune extension de fichier n'est présente, Athena traite les données comme du texte brut non compressé. Si vos données sont compressées, assurez-vous que le nom de fichier comprend l'extension de compression, par exemple `gz`.
- Le format de fichier ZIP n'est pas pris en charge.
- Pour interroger les journaux Amazon Data Firehose depuis Athena, les formats pris en charge incluent la compression GZIP ou les fichiers ORC avec compression SNAPPY.

- Pour plus d'informations sur l'utilisation de la compression, consultez la section 3 (« Compresser et diviser des fichiers ») du billet de blog AWS Big Data sur les 10 meilleurs conseils d'optimisation des performances pour Amazon Athena.

Prise en charge de la compression de la table Hive par format de fichier

La prise en charge de la compression Hive dans Athena dépend de la version du moteur.

Prise en charge de la compression Hive dans la version 3 du moteur Athena

Le tableau suivant résume la prise en charge des formats de compression dans la version 3 du moteur Athena pour les formats de fichier de stockage dans Apache Hive. Le format de fichier texte inclut TSV, CSV, JSON et SerDe personnalisés pour le texte. La mention « Oui » ou « Non » dans une cellule s'applique de la même manière aux opérations de lecture et d'écriture, sauf indication contraire. Pour les besoins de cette table, les instructions CREATE TABLE, CTAS et INSERT INTO sont considérées comme des opérations d'écriture. Pour plus d'informations sur l'utilisation des niveaux de compression ZSTD sur Athena, consultez Utilisation des niveaux de compression ZSTD dans Athena.

	Avro	Ion	ORC	Parquet	Fichier texte
BZIP2	Oui	Oui	Non	Non	Oui
DEFLATE	Oui	Non	Non	Non	Non
GZIP	Non	Oui	Non	Oui	Oui
LZ4	Non	Oui	Oui	Oui	Oui
LZO	Non	Écriture - Non Lecture - Oui	Non	Oui	Écriture - Non Lecture - Oui
SNAPPY	Oui	Oui	Oui	Oui	Oui
ZLIB	Non	Non	Oui	Non	Non

	Avro	Ion	ORC	Parquet	Fichier texte
ZSTD	Oui	Oui	Oui	Oui	Oui
NONE	Oui	Oui	Oui	Oui	Oui

Prise en charge de la compression dans la version 2 du moteur Athena

Le tableau suivant résume la prise en charge des formats de compression dans la version 2 du moteur Athena pour Apache Hive. Le format de fichier texte inclut TSV, CSV, JSON et SerDe personnalisés pour le texte. La mention « Oui » ou « Non » dans une cellule s'applique de la même manière aux opérations de lecture et d'écriture, sauf indication contraire. Pour les besoins de cette table, les instructions CREATE TABLE, CTAS et INSERT INTO sont considérées comme des opérations d'écriture.

	Avro	Ion	ORC	Parquet	Fichier texte
BZIP2	Oui	Oui	Non	Non	Oui
DEFLATE	Oui	Non	Non	Non	Non
GZIP	Non	Oui	Non	Oui	Oui
LZ4	Non	Non	Oui	Écriture - Oui Lecture - Non	Écriture - Non Lecture - Oui
LZO	Non	Écriture - Non Lecture - Oui	Non	Oui	Écriture - Non Lecture - Oui
SNAPPY	Oui	Oui	Oui	Oui	Oui

	Avro	Ion	ORC	Parquet	Fichier texte
ZLIB	Non	Non	Oui	Non	Non
ZSTD	Non	Oui	Oui	Oui	Oui
NONE	Oui	Oui	Oui	Oui	Oui

Prise en charge de la compression de la table Iceberg par format de fichier

La prise en charge de la compression Apache Iceberg dans Athena dépend de la version du moteur.

Prise en charge de la compression Iceberg dans la version 3 du moteur Athena

Le tableau suivant résume la prise en charge des formats de compression dans la version 3 du moteur Athena pour les formats de fichier de stockage dans Apache Iceberg. La mention « Oui » ou « Non » dans une cellule s'applique de la même manière aux opérations de lecture et d'écriture, sauf indication contraire. Pour les besoins de cette table, les instructions CREATE TABLE, CTAS et INSERT INTO sont considérées comme des opérations d'écriture. Le format de stockage par défaut pour Iceberg dans la version 3 du moteur Athena est Parquet. Le format de compression par défaut pour Iceberg dans la version 3 du moteur Athena est ZSTD. Pour plus d'informations sur l'utilisation des niveaux de compression ZSTD sur Athena, consultez Utilisation des niveaux de compression ZSTD dans Athena.

	Avro	ORC	Parquet (par défaut)
BZIP2	Non	Non	Non
GZIP	Oui	Non	Oui
LZ4	Non	Oui	Non
SNAPPY	Oui	Oui	Oui
ZLIB	Non	Oui	Non
ZSTD	Oui	Oui	Oui (par défaut)

	Avro	ORC	Parquet (par défaut)
NONE	Oui (précisez `None` ou `Deflate`)	Oui	Oui (précisez `None` ou `Uncompressed`)

Prise en charge de la compression Iceberg dans la version 2 du moteur Athena

Le tableau suivant résume la prise en charge des formats de compression dans la version 2 du moteur Athena pour Apache Iceberg. La mention « Oui » ou « Non » dans une cellule s'applique de la même manière aux opérations de lecture et d'écriture, sauf indication contraire. Pour les besoins de cette table, les instructions CREATE TABLE, CTAS et INSERT INTO sont considérées comme des opérations d'écriture. Le format de stockage par défaut pour Iceberg dans la version 2 du moteur Athena est Parquet. Le format de compression par défaut pour Iceberg dans la version 2 du moteur Athena est GZIP.

	Avro (Non pris en charge)	ORC (Non pris en charge)	Parquet (par défaut)
BZIP2	Non	Non	Non
GZIP	Non	Non	Oui (par défaut)
LZ4	Non	Non	Non
SNAPPY	Non	Non	Oui
ZLIB	Non	Non	Non
ZSTD	Non	Non	Oui
NONE	Non	Non	Oui

Utilisation des niveaux de compression ZSTD dans Athena

L’algorithme de compression de données en temps réel Zstandard est un algorithme de compression rapide qui fournit des taux de compression élevés. La bibliothèque Zstandard est un logiciel open

source qui utilise une licence BSD. Athena prend en charge la lecture et l'écriture de données ORC, Parquet et de fichiers texte compressés selon la norme ZSTD.

Vous pouvez utiliser les niveaux de compression ZSTD pour ajuster le taux et la vitesse de compression en fonction de vos besoins. La bibliothèque ZSTD prend en charge des niveaux de compression compris entre 1 et 22. Athena utilise le niveau de compression ZSTD 3 par défaut.

Les niveaux de compression offrent des compromis précis entre la vitesse de compression et le niveau de compression atteint. Des niveaux de compression plus faibles offrent une vitesse plus importante, mais des fichiers de plus grande taille. Par exemple, vous pouvez utiliser le niveau 1 si la vitesse est la plus importante et le niveau 22 si la taille est la plus importante. Le niveau 3 convient à de nombreux cas d'utilisation et constitue le niveau par défaut. Utilisez les niveaux supérieurs à 19 avec prudence, car ils nécessitent plus de mémoire. La bibliothèque ZSTD propose également des niveaux de compression négatifs qui étendent la plage de vitesses et de taux de compression. Pour plus d'informations, consultez le RFC de compression Zstandard.

L'abondance de niveaux de compression offre de nombreuses possibilités de réglage précis. Toutefois, assurez-vous de mesurer vos données et de prendre en compte les compromis lorsque vous décidez d'un niveau de compression. Nous vous recommandons d'utiliser le niveau 3 par défaut ou un niveau compris entre 6 et 9 pour obtenir un compromis raisonnable entre la vitesse de compression et la taille des données compressées. Réservez les niveaux 20 et plus pour les cas où la taille est la plus importante et où la vitesse de compression n'est pas un problème.

Considérations et restrictions

Lorsque vous utilisez le niveau de compression ZSTD dans Athena, tenez compte des points suivants.

- La propriété `compression_level` ZSTD est prise en charge uniquement dans la version 3 du moteur Athena.
- La propriété `compression_level` ZSTD est prise en charge pour les instructions `ALTER TABLE`, `CREATE TABLE`, `CREATE TABLE AS` (CTAS) et `UNLOAD`.
- La propriété `compression_level` est facultative.
- La propriété `compression_level` est prise en charge uniquement pour la compression ZSTD.
- Les niveaux de compression possibles sont compris entre 1 et 22.
- Le niveau de compression par défaut est le niveau 3.

Pour de plus amples informations sur la prise en charge de la compression ZSTD Apache Hive dans Athena, consultez Prise en charge de la compression de la table Hive par format de fichier. Pour de plus amples informations sur la prise en charge de la compression ZSTD Apache Iceberg dans Athena, consultez Prise en charge de la compression de la table Iceberg par format de fichier.

Spécification des niveaux de compression ZSTD

Pour spécifier le niveau de compression ZSTD pour les instructions `ALTER TABLE`, `CREATE TABLE`, `CREATE TABLE AS` et `UNLOAD`, utilisez la propriété `compression_level`. Pour spécifier la compression ZSTD elle-même, vous devez utiliser la propriété de compression individuelle utilisée par la syntaxe de l'instruction.

ALTER TABLE SET TBLPROPERTIES

Dans la clause `SET TBLPROPERTIES` de l'instruction ALTER TABLE SET TBLPROPERTIES, spécifiez la compression ZSTD à l'aide de `'write.compression' = ' ZSTD'` ou de `'parquet.compression' = 'ZSTD'`. Utilisez ensuite la propriété `compression_level` pour spécifier une valeur comprise entre 1 et 22 (par exemple, '`compression_level' = 5`). Si vous ne spécifiez aucune propriété de niveau de compression, le niveau de compression est défini par défaut sur 3.

Exemple

L'exemple suivant modifie la table `existing_table` pour utiliser le format de fichier Parquet avec une compression ZSTD et un niveau de compression ZSTD 4. Notez que la valeur du niveau de compression doit être saisie sous la forme d'une chaîne plutôt que d'un entier.

```
ALTER TABLE existing_table
SET TBLPROPERTIES ('parquet.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE TABLE

Dans la clause `TBLPROPERTIES` de l'instruction CREATE TABLE, spécifiez '`write.compression' = 'ZSTD'` ou `'parquet.compression' = 'ZSTD'`, puis utilisez `compression_level = compression_level` et spécifiez une valeur comprise entre 1 et 22. Si la propriété `compression_level` n'est pas spécifiée, le niveau de compression par défaut est 3.

Exemple

L'exemple suivant crée un tableau au format de fichier Parquet à l'aide de la compression ZSTD et du niveau de compression ZSTD 4.

```
CREATE EXTERNAL TABLE new_table (
  `col0` string COMMENT '',
  `col1` string COMMENT ''
)
STORED AS PARQUET
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
TBLPROPERTIES ('write.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE TABLE AS (CTAS)

Dans la clause `WITH` de l'instruction CREATE TABLE AS, spécifiez `write_compression = 'ZSTD'` ou`parquet_compression = 'ZSTD'`, puis utilisez `compression_level = compression_level` et spécifiez une valeur comprise entre 1 et 22. Si la propriété `compression_level` n'est pas spécifiée, le niveau de compression par défaut est 3.

Exemple

L'exemple CTAS suivant spécifie Parquet comme format de fichier utilisant la compression ZSTD avec un niveau de compression 4.

```
CREATE TABLE new_table
WITH ( format = 'PARQUET', write_compression = 'ZSTD', compression_level = 4)
AS SELECT * FROM old_table
```

UNLOAD

Dans la clause `WITH` de l'instruction UNLOAD, spécifiez `compression = 'ZSTD'`, puis utilisez `compression_level = compression_level` et spécifiez une valeur comprise entre 1 et 22. Si la propriété `compression_level` n'est pas spécifiée, le niveau de compression par défaut est 3.

Exemple

L'exemple suivant décharge les résultats de la requête vers l'emplacement spécifié à l'aide du format de fichier Parquet, de la compression ZSTD et du niveau de compression ZSTD 4.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET', compression = 'ZSTD', compression_level = 4)
```

Étiquetage des ressources Athena

une identification est constituée d'une clé et d'une valeur que vous définissez. Lorsque vous étiquetez une ressource Athena, vous lui attribuez des métadonnées personnalisées. Vous pouvez utiliser des étiquettes pour classer vos ressources AWS de différentes manières, par exemple, par objectif, par propriétaire ou par environnement. Dans Athena, les ressources telles que les groupes de travail, les catalogues de données et les réserves de capacité sont des ressources étiquetables. Par exemple, vous pouvez créer un ensemble d'identifications pour les groupes de travail de votre compte, afin de vous aider à suivre les propriétaires de groupes de travail ou à identifier les groupes de travail grâce à leur objectif. Si vous activez également les balises en tant que balises de répartition des coûts dans la console de Billing and Cost Management, les coûts associés à l'exécution de requêtes apparaissent dans votre rapport sur les coûts et l'utilisation avec cette balise de répartition des coûts. Nous vous recommandons d'utiliser les bonnes pratiques de balisage AWS pour créer un ensemble d'identifications cohérent capable de répondre aux exigences de votre organisation.

Vous pouvez utiliser les étiquettes à l'aide de la console Athena ou des opérations d'API.

Rubriques

Principes de base des étiquettes

L'étiquette est une identification que vous attribuez à une ressource Athena. Chaque identification est constituée d'une clé et d'une valeur facultative que vous définissez.

Les étiquettes vous permettent de catégoriser vos ressources AWS de différentes manières. Par exemple, vous pouvez définir un ensemble d'identifications pour les groupes de travail de votre compte vous permettant de suivre chaque propriétaire ou objectif de groupe de travail.

Vous pouvez ajouter des étiquettes lors de la création d'un nouveau groupe de travail ou catalogue de données Athena, ou vous pouvez ajouter, modifier ou supprimer des étiquettes de ceux-ci. Vous pouvez modifier une identification dans la console. Si vous utilisez les opérations d'API, pour

modifier une identification, supprimez l'ancienne et ajoutez-en une nouvelle. Si vous supprimez une ressource, les identifications associées à celle-ci seront également supprimées.

Athena n'attribue pas automatiquement d'étiquettes à vos ressources. Vous pouvez modifier les clés et valeurs d'identification, et vous pouvez retirer des identifications d'une ressource à tout moment. Vous pouvez définir la valeur d'une identification sur une chaîne vide, mais vous ne pouvez pas définir la valeur d'une identification sur null. N'ajoutez pas de clés d'identification en double à la même ressource. Dans ce cas, Athena envoie un message d'erreur. Si vous utilisez l'action TagResource pour identifier une ressource à l'aide d'une clé d'identification existante, la nouvelle valeur d'identification remplace l'ancienne valeur.

Dans IAM, vous pouvez contrôler quels utilisateurs de votre compte Amazon Web Services disposent d'autorisations de créer, modifier et supprimer ou répertorier des étiquettes. Pour plus d'informations, consultez Politiques de contrôle d'accès IAM basées sur les étiquettes.

Pour obtenir liste complète des actions des étiquettes Amazon Athena, consultez les noms des actions de l'API dans la rubrique Référence d'API Amazon Athena.

Vous pouvez utiliser des identifications pour la facturation. Pour plus d'informations, consultez la rubrique Utilisation d'identifications pour la facturation dans le guide de l'utilisateur AWS Billing and Cost Management.

Pour plus d'informations, consultez Restrictions liées aux étiquettes.

Restrictions liées aux étiquettes

les identifications disposent des restrictions suivantes :

- Dans Athena, vous pouvez étiqueter des groupes de travail et des catalogues de données. Vous ne pouvez pas identifier de requêtes.
- Le nombre maximum d'identifications par ressource est de 50. Pour rester dans la limite, vérifiez et supprimez les identifications non utilisées.
- Pour chaque ressource, chaque clé d'identification doit être unique, et chaque clé d'identification peut avoir une seule valeur. N'ajoutez pas simultanément de clés d'identification dupliquées à la même ressource. Dans ce cas, Athena envoie un message d'erreur. Si vous identifiez une ressource à l'aide d'une clé d'identification existante en exécutant une action `TagResource` distincte, la nouvelle valeur d'identification remplace l'ancienne.
- La longueur de la clé d'identification est comprise entre 1 et 128 caractères Unicode en UTF-8

- La longueur de la valeur d'identification est comprise entre 0 et 256 caractères Unicode en UTF-8

  Les opérations de balisage, telles que l'ajout, la modification, la suppression ou la liste des identifications, exigent que vous spécifiez un ARN pour la ressource de groupe de travail.
- Athena vous permet d'utiliser des lettres, des chiffres, des espaces représentés en UTF-8, ainsi que les caractères suivants : + - = . _ : / @.
- Les clés et valeurs d'identification sont sensibles à la casse.
- Le préfixe `"aws:"` des clés d'identification est réservé à l'utilisation par AWS. Vous ne pouvez pas modifier ou supprimer des clés d'identification ayant ce préfixe. les identifications ayant ce préfixe ne sont pas comptabilisées dans votre limite d'identifications par ressource.
- Les étiquettes que vous attribuez sont disponibles uniquement pour votre compte Amazon Web Services.

Utilisation des identifications sur des groupes de travail dans la console

À l'aide de la console Athena, vous pouvez voir quelles étiquettes sont en cours d'utilisation pour chaque groupe de travail de votre compte. Vous pouvez afficher des identifications uniquement par le groupe de travail. Vous pouvez également utiliser la console Athena pour appliquer, modifier ou supprimer des étiquettes dans un seul groupe de travail à la fois.

Vous pouvez rechercher des groupes de travail à l'aide des identifications créées.

Rubriques

- Affichage des identifications pour des groupes de travail individuels
- Ajout et suppression d'identifications sur un groupe de travail individuel

Affichage des identifications pour des groupes de travail individuels

Affichage des identifications d'un groupe de travail individuel dans la console Athena

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le menu de navigation, choisissez Workgroups (Groupes de travail), puis choisissez le groupe de travail souhaité.
4. Effectuez l'une des actions suivantes :

   - Sélectionnez l'onglet Tags (Identifications). Si la liste des identifications est longue, utilisez la zone de recherche.
   - Choisissez Edit (Modifier), puis faites défiler jusqu'à la section Tags (Identifications).

Ajout et suppression d'identifications sur un groupe de travail individuel

Vous pouvez gérer les identifications d'un groupe de travail individuel directement à partir de l'onglet Workgroups (Groupes de travail).

> **Note**
>
> Si vous souhaitez que les utilisateurs ajoutent des étiquettes lorsqu'ils créent un groupe de travail dans la console ou transmettent des étiquettes lorsqu'ils utilisent l'action CreateWorkGroup, assurez-vous que vous accordez aux utilisateurs IAM des autorisations pour les actions TagResource et CreateWorkGroup.

Ajout d'une identification lors de la création d'un nouveau groupe de travail

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans le menu de navigation, choisissez Workgroups (Groupes de travail).

3. Choisissez Create workgroup (Créer un groupe de travail) et saisissez les valeurs nécessaires. Pour obtenir des instructions complètes, consultez Créer un groupe de travail.
4. Dans la section Tags (Identifications), ajoutez une ou plusieurs identifications en spécifiant les clés et les valeurs. N'ajoutez pas simultanément de clés d'identification dupliquées au même groupe de travail. Dans ce cas, Athena envoie un message d'erreur. Pour plus d'informations, consultez Restrictions liées aux étiquettes.
5. Une fois que vous avez terminé, choisissez Create workgroup (Créer un groupe de travail).

Pour ajouter une identification à un groupe de travail existant ou la modifier

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans le panneau de navigation, choisissez Workgroups (Groupes de travail).
3. Choisissez le groupe de travail à modifier.
4. Effectuez l'une des actions suivantes :
   - Choisissez l'onglet Tags (Identifications), puis Manage tags (Gérer les identifications).
   - Choisissez Edit (Modifier), puis faites défiler jusqu'à la section Tags (Identifications).
5. Spécifiez une clé et une valeur pour chaque identification. Pour plus d'informations, consultez Restrictions liées aux étiquettes.
6. Choisissez Enregistrer.

Pour supprimer une identification d'un groupe de travail individuel

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Dans le panneau de navigation, choisissez Workgroups (Groupes de travail).
3. Choisissez le groupe de travail à modifier.
4. Effectuez l'une des actions suivantes :
   - Choisissez l'onglet Tags (Identifications), puis Manage tags (Gérer les identifications).
   - Choisissez Edit (Modifier) puis faites défiler jusqu'à la section Tags (Identifications).
5. Dans la liste des identifications, choisissez Remove (Supprimer) pour l'identification à supprimer, puis choisissez Save (Enregistrer).

Utilisation des opérations d'identification

Utilisez les opérations d'identification suivantes pour ajouter, supprimer ou répertorier des identifications sur une ressource.

API	INTERFACE DE LIGNE DE COMMANDE (CLI)	Description de l'action
`TagResource`	`tag-resource`	Ajoutez ou remplacez une ou plusieurs identifications sur la ressource où l'ARN est spécifié.
`UntagResource`	`untag-resource`	Supprimez une ou plusieurs identifications de la ressource ayant l'ARN spécifié.
`ListTagsForResource`	`list-tags-for-resource`	Répertorie une ou plusieurs identifications pour la ressource qui a l'ARN spécifié.

Ajout d'identifications lors de la création d'une ressource

Pour ajouter des identifications lorsque vous créez un groupe de travail ou un catalogue de données, utilisez le paramètre `tags` avec les opérations d'API `CreateWorkGroup` ou `CreateDataCatalog` API ou avec les commandes AWS CLI, `create-work-group` ou `create-data-catalog`.

Gestion des identifications à l'aide des opérations d'API

Les exemples de cette section montrent comment utiliser les opérations d'API d'identification pour gérer les identifications sur les groupes de travail et les catalogues de données. Les exemples sont dans le langage de programmation Java.

Example TagResource

L'exemple suivant ajoute deux identifications au groupe de travail `workgroupA`:

```
List<Tag> tags = new ArrayList<>();
tags.add(new Tag().withKey("tagKey1").withValue("tagValue1"));
```

```
tags.add(new Tag().withKey("tagKey2").withValue("tagValue2"));

TagResourceRequest request = new TagResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA")
    .withTags(tags);

client.tagResource(request);
```

L'exemple suivant ajoute deux identifications au catalogue de données `datacatalogA`:

```
List<Tag> tags = new ArrayList<>();
tags.add(new Tag().withKey("tagKey1").withValue("tagValue1"));
tags.add(new Tag().withKey("tagKey2").withValue("tagValue2"));

TagResourceRequest request = new TagResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA")
    .withTags(tags);

client.tagResource(request);
```

> **Note**
>
> N'ajoutez pas de clés d'identification en double à la même ressource. Dans ce cas, Athena envoie un message d'erreur. Si vous identifiez une ressource à l'aide d'une clé d'identification existante en exécutant une action `TagResource` distincte, la nouvelle valeur d'identification remplace l'ancienne.

Example UntagResource

L'exemple suivant supprime `tagKey2` du groupe de travail `workgroupA` :

```
List<String> tagKeys = new ArrayList<>();
tagKeys.add("tagKey2");

UntagResourceRequest request = new UntagResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA")
    .withTagKeys(tagKeys);

client.untagResource(request);
```

L'exemple suivant supprime `tagKey2` du catalogue de données `datacatalogA` :

```
List<String> tagKeys = new ArrayList<>();
tagKeys.add("tagKey2");

UntagResourceRequest request = new UntagResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA")
    .withTagKeys(tagKeys);

client.untagResource(request);
```

Example ListTagsForResource

L'exemple suivant répertorie les identifications du groupe de travail `workgroupA`:

```
ListTagsForResourceRequest request = new ListTagsForResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA");

ListTagsForResourceResult result = client.listTagsForResource(request);

List<Tag> resultTags = result.getTags();
```

L'exemple suivant répertorie les identifications du catalogue de données `datacatalogA`:

```
ListTagsForResourceRequest request = new ListTagsForResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA");

ListTagsForResourceResult result = client.listTagsForResource(request);

List<Tag> resultTags = result.getTags();
```

Gestion des balises à l'aide du kit AWS CLI

Les sections suivantes montrent comment utiliser AWS CLI pour créer et gérer des identifications dans les catalogues de données.

Ajout d'identifications à une ressource : tag-resource

La commande `tag-resource` ajoute une ou plusieurs identifications à une ressource spécifiée.

Syntaxe

```
aws athena tag-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name --tags
Key=string,Value=string Key=string,Value=string
```

Le paramètre `--resource-arn` spécifie la ressource à laquelle les identifications sont ajoutées. Le paramètre `--tags` spécifie une liste de paires clé-valeur séparées par des espaces à ajouter en tant qu'identifications à la ressource.

Example

L'exemple suivant ajoute des identifications au catalogue de données `mydatacatalog`.

```
aws athena tag-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog --tags Key=Color,Value=Orange
 Key=Time,Value=Now
```

Pour afficher le résultat, utilisez la commande `list-tags-for-resource`.

Pour plus d'informations sur l'ajout de balises lors de l'utilisation de la commande `create-data-catalog`, consultez Enregistrer un catalogue : C reate-data-catalog.

Liste des identifications d'une ressource : list-tags-for-resource

La commande `list-tags-for-resource` répertorie les identifications de la ressource spécifiée.

Syntaxe

```
aws athena list-tags-for-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name
```

Le paramètre `--resource-arn` spécifie la ressource pour laquelle les identifications sont répertoriées.

L'exemple suivant répertorie les identifications du catalogue de données `mydatacatalog`.

```
aws athena list-tags-for-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog
```

L'exemple de résultat suivant est au format JSON.

```
{
    "Tags": [
        {
            "Key": "Time",
            "Value": "Now"
        },
        {
            "Key": "Color",
            "Value": "Orange"
        }
    ]
}
```

Suppression d'identifications d'une ressource : untag-resource

La commande `untag-resource` supprime les clés d'identification spécifiées et leurs valeurs associées provenant de la ressource spécifiée.

Syntaxe

```
aws athena untag-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name --tag-keys
key_name [key_name ...]
```

Le paramètre `--resource-arn` spécifie la ressource à partir de laquelle les identifications sont supprimées. Le paramètre `--tag-keys` prend une liste de noms de clés séparés par des espaces. Pour chaque nom de clé spécifié, la commande `untag-resource` supprime à la fois la clé et sa valeur.

L'exemple suivant supprime les clés `Color` et `Time`, et leurs valeurs issues de la ressource de catalogue `mydatacatalog`.

```
aws athena untag-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog --tag-keys Color Time
```

Politiques de contrôle d'accès IAM basées sur les étiquettes

Le fait de disposer d'étiquettes vous permet d'écrire une politique IAM qui inclut le bloc `Condition` permettant de contrôler l'accès à une ressource en fonction de ses étiquettes.

Exemples de politique d'identification pour les groupes de travail

Example 1. politique d'étiquetage de base

La politique IAM suivante vous permet d'exécuter des requêtes et d'interagir avec des étiquettes pour le groupe de travail nommé `workgroupA` :

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Effect": "Allow",
            "Action": [
                "athena:ListWorkGroups",
                "athena:ListEngineVersions",
                "athena:ListDataCatalogs",
                "athena:ListDatabases",
                "athena:GetDatabase",
                "athena:ListTableMetadata",
                "athena:GetTableMetadata"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:GetWorkGroup",
                "athena:TagResource",
                "athena:UntagResource",
                "athena:ListTagsForResource",
                "athena:StartQueryExecution",
                "athena:GetQueryExecution",
                "athena:BatchGetQueryExecution",
                "athena:ListQueryExecutions",
                "athena:StopQueryExecution",
                "athena:GetQueryResults",
                "athena:GetQueryResultsStream",
                "athena:CreateNamedQuery",
                "athena:GetNamedQuery",
                "athena:BatchGetNamedQuery",
                "athena:ListNamedQueries",
                "athena:DeleteNamedQuery",
                "athena:CreatePreparedStatement",
                "athena:GetPreparedStatement",
```

```
                "athena:ListPreparedStatements",
                "athena:UpdatePreparedStatement",
                "athena:DeletePreparedStatement"
            ],
            "Resource": "arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA"
        }
    ]
}
```

Example 2 : bloc de politique qui refuse des actions sur un groupe de travail basé sur une clé d'identification et une paire de valeur d'identification

Les identifications associées à un groupe de travail existant sont appelées identifications de ressource. les identifications de ressources vous permettent d'écrire des blocs de politique comme les suivants qui refusent les actions répertoriées sur un groupe de travail marqué avec une paire clé-valeur comme `stack`, `production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "athena:GetWorkGroup",
                "athena:UpdateWorkGroup",
                "athena:DeleteWorkGroup",
                "athena:TagResource",
                "athena:UntagResource",
                "athena:ListTagsForResource",
                "athena:StartQueryExecution",
                "athena:GetQueryExecution",
                "athena:BatchGetQueryExecution",
                "athena:ListQueryExecutions",
                "athena:StopQueryExecution",
                "athena:GetQueryResults",
                "athena:GetQueryResultsStream",
                "athena:CreateNamedQuery",
                "athena:GetNamedQuery",
                "athena:BatchGetNamedQuery",
                "athena:ListNamedQueries",
                "athena:DeleteNamedQuery",
                "athena:CreatePreparedStatement",
                "athena:GetPreparedStatement",
```

```
                "athena:ListPreparedStatements",
                "athena:UpdatePreparedStatement",
                "athena:DeletePreparedStatement"
            ],
            "Resource": "arn:aws:athena:us-east-1:123456789012:workgroup/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stack": "production"
                }
            }
        }
    ]
}
```

Example 3. bloc de politique qui limite les demandes d'action de changement d'identifications aux identifications spécifiées

Les identifications qui sont transmises en tant que paramètres aux opérations qui modifient les identifications (par exemple `TagResource`, `UntagResource` ou `CreateWorkGroup` avec des identifications) sont appelées identifications de requête. L'exemple de bloc de politique suivant n'autorise l'opération `CreateWorkGroup` que si l'une des identifications passées a la clé `costcenter` et la valeur 1, 2 ou 3.

> **Note**
>
> Si vous voulez permettre à un rôle IAM de transmettre des identifications dans le cadre d'une opération `CreateWorkGroup`, assurez-vous d'accorder au rôle des autorisations pour les actions `TagResource` et `CreateWorkGroup`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateWorkGroup",
                "athena:TagResource"
            ],
            "Resource": "arn:aws:athena:us-east-1:123456789012:workgroup/*",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:RequestTag/costcenter": [
                        "1",
                        "2",
                        "3"
                    ]
                }
            }
        }
    ]
}
```

Exemples de politique d'identification pour les catalogues de données

Example 1. politique d'étiquetage de base

La politique IAM suivante vous permet d'interagir avec les étiquettes pour le catalogue de données nommé `datacatalogA` :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListWorkGroups",
                "athena:ListEngineVersions",
                "athena:ListDataCatalogs",
                "athena:ListDatabases",
                "athena:GetDatabase",
                "athena:ListTableMetadata",
                "athena:GetTableMetadata"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:GetWorkGroup",
                "athena:TagResource",
                "athena:UntagResource",
                "athena:ListTagsForResource",
                "athena:StartQueryExecution",
```

```
                "athena:GetQueryExecution",
                "athena:BatchGetQueryExecution",
                "athena:ListQueryExecutions",
                "athena:StopQueryExecution",
                "athena:GetQueryResults",
                "athena:GetQueryResultsStream",
                "athena:CreateNamedQuery",
                "athena:GetNamedQuery",
                "athena:BatchGetNamedQuery",
                "athena:ListNamedQueries",
                "athena:DeleteNamedQuery"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateDataCatalog",
                "athena:GetDataCatalog",
                "athena:UpdateDataCatalog",
                "athena:DeleteDataCatalog",
                "athena:ListDatabases",
                "athena:GetDatabase",
                "athena:ListTableMetadata",
                "athena:GetTableMetadata",
                "athena:TagResource",
                "athena:UntagResource",
                "athena:ListTagsForResource"
            ],
            "Resource": "arn:aws:athena:us-
east-1:123456789012:datacatalog/datacatalogA"
        }
    ]
}
```

Example 2 : bloc de politique qui refuse des actions sur un groupe de travail basé sur une paire clé d'identification/valeur d'identification

Vous pouvez utiliser des identifications de ressource pour écrire des blocs de politique qui refusent des actions spécifiques sur les catalogues de données qui sont balisés avec des paires clé-valeur

d'identification spécifiques. L'exemple de politique suivant refuse les actions sur les catalogues de données qui ont la paire clé-valeur d'identification `stack`, `production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "athena:CreateDataCatalog",
                "athena:GetDataCatalog",
                "athena:UpdateDataCatalog",
                "athena:DeleteDataCatalog",
                "athena:GetDatabase",
                "athena:ListDatabases",
                "athena:GetTableMetadata",
                "athena:ListTableMetadata",
                "athena:StartQueryExecution",
                "athena:TagResource",
                "athena:UntagResource",
                "athena:ListTagsForResource"
            ],
            "Resource": "arn:aws:athena:us-east-1:123456789012:datacatalog/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stack": "production"
                }
            }
        }
    ]
}
```

Example 3. bloc de politique qui limite les demandes d'action de changement d'identifications aux identifications spécifiées

Les identifications qui sont transmises en tant que paramètres aux opérations qui modifient les identifications (par exemple `TagResource`, `UntagResource` ou `CreateDataCatalog` avec des identifications) sont appelées identifications de requête. L'exemple de bloc de politique suivant n'autorise l'opération `CreateDataCatalog` que si l'une des identifications passées a la clé `costcenter` et la valeur `1`, `2` ou `3`.

> **Note**
>
> Si vous voulez permettre à un rôle IAM de transmettre des identifications dans le cadre d'une opération `CreateDataCatalog`, assurez-vous d'accorder au rôle des autorisations pour les actions `TagResource` et `CreateDataCatalog`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateDataCatalog",
                "athena:TagResource"
            ],
            "Resource": "arn:aws:athena:us-east-1:123456789012:datacatalog/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/costcenter": [
                        "1",
                        "2",
                        "3"
                    ]
                }
            }
        }
    ]
}
```

Service Quotas

> **Note**
>
> La console Service Quotas fournit des informations sur les quotas dans Amazon Athena. Vous pouvez également utiliser la console Service Quotas pour demander des augmentations de quota pour les quotas ajustables. Pour connaître les limites de schéma associées à AWS Glue , consultez la page Points de terminaison et quotas AWS Glue. Pour

des informations générales sur les quotas AWS de service, consultez la section sur les quotas de AWS service dans le Références générales AWS.

Requêtes

Votre compte dispose des quotas de requête par défaut suivants pour Amazon Athena. Pour obtenir des détails, consultez la page Points de terminaison et quotas Amazon Athena de la Références générales AWS.

- Requêtes DDL actives – Nombre de requêtes DDL actives. Les requêtes DDL comprennent les requêtes `CREATE TABLE` et `ALTER TABLE ADD PARTITION`.
- Expiration d'une requête DDL – Durée maximale, en minutes, pendant laquelle une requête DDL peut s'exécuter avant son annulation.
- Requêtes DML actives – Nombre de requêtes DML actives. Les requêtes DML comprennent `SELECT`, `CREATE TABLE AS` (CTAS) et des requêtes `INSERT INTO`. Les quotas spécifiques varient selon la région AWS .
- Expiration d'une requête DML – Durée maximale, en minutes, pendant laquelle une requête DML peut s'exécuter avant son annulation. Vous pouvez demander une augmentation de ce délai d'attente jusqu'à un maximum de 240 minutes.

Pour demander des augmentations de quota, vous pouvez utiliser la console Service Quotas Athena.

Athena traite les requêtes en affectant des ressources en se basant sur la charge globale du service et le nombre de demandes entrantes. Vos requêtes peuvent être temporairement mises en attente avant d'être exécutées. Les processus asynchrones récupèrent les requêtes dans les files d'attente et les exécutent sur les ressources physiques dès que celles-ci sont disponibles et aussi longtemps que la configuration de votre compte le permet.

Un quota de requêtes DML ou DDL inclut à la fois les requêtes en cours d'exécution et en file d'attente. Par exemple, si votre quota de requêtes DML est de 25 et que le nombre total de requêtes en cours et en file d'attente est de 26, la requête 26 provoquera une erreur. TooManyRequestsException

> **Note**
>
> Si vous souhaitez contrôler directement la simultanéité des requêtes que vous exécutez dans Athena, vous pouvez utiliser les réserves de capacité. Pour plus d'informations, consultez Gestion de la capacité de traitement des requêtes.

Longueur de chaîne de requête

La longueur de chaîne de requête maximum autorisée est de 262 144 octets, où les chaînes sont encodées en UTF-8. Il ne s'agit pas d'un quota ajustable. Toutefois, vous pouvez contourner cette limitation en divisant les longues requêtes en plusieurs petites requêtes. Pour plus d'informations, consultez la rubrique Comment augmenter la longueur maximale de la chaîne de requête dans Athena ? dans le Centre de connaissances AWS .

Groupes de travail

Lorsque vous travaillez avec des groupes de travail Athena, n'oubliez pas les points suivants :

- Les Service Quotas (Service Quotas) Athena sont partagés entre tous les groupes de travail d'un compte.
- Le nombre maximal de groupes de travail que vous pouvez créer par région dans votre compte est de 1 000.
- Le nombre maximal d'instructions préparées dans un groupe de travail est de 1 000.
- Le nombre maximum d'identifications par groupe de travail est de 50. Pour plus d'informations, consultez Restrictions liées aux étiquettes.

Bases de données, tables et partitions

- Si vous utilisez le AWS Glue Data Catalog avec Athena, consultez la section AWS Glue Points de terminaison et quotas pour les quotas de service sur les tables, les bases de données et les partitions, par exemple le nombre maximum de bases de données ou de tables par compte.
  - Bien qu'Athena prenne en charge l'interrogation de AWS Glue tables contenant 10 millions de partitions, Athena ne peut pas lire plus d'un million de partitions en un seul scan.
- Si vous n'en utilisez pas AWS Glue Data Catalog, le nombre de partitions par table est de 20 000. Vous pouvez demander une augmentation de quota.

Compartiments Amazon S3

Lorsque vous travaillez avec des compartiments Simple Storage Service (Amazon S3), rappelez-vous les points suivants :

- Simple Storage Service (Amazon S3) a un quota de service par défaut de 100 compartiments par compte.
- Athena a également besoin d'un compartiment distinct pour journaliser les résultats.
- Vous pouvez demander une augmentation de quota allant jusqu'à 1 000 compartiments Simple Storage Service (Amazon S3) par compte AWS .

Quotas d'appel d'API par compte

Les API Athena ont les quotas par défaut pour le nombre d'appels à l'API par compte (et non par requête) :

Nom d'API	Nombre d'appels par défaut par seconde	Capacité de débordement
`BatchGetNamedQuery` , `ListNamedQueries` , `ListQueryExecutions`	5	jusqu'à 10
`CreateNamedQuery` , `DeleteNamedQuery` , `GetNamedQuery`	5	jusqu'à 20
`BatchGetQueryExecution`	20	jusqu'à 40
`StartQueryExecution` , `StopQueryExecution`	20	jusqu'à 80
`GetQueryExecution` , `GetQueryResults`	100	jusqu'à 200

Par exemple, vous pouvez prendre jusqu'à 20 appels par seconde pour `StartQueryExecution`. En outre, si cette API n'est pas appelée pendant 4 secondes, votre compte accumule une capacité de transmission en mode rafale allant jusqu'à 80 appels. Dans ce cas, votre application peut effectuer jusqu'à 80 appels de cette API en mode rafale.

Si vous utilisez l'une de ces API et que vous dépassez le quota par défaut pour le nombre d'appels par seconde ou la capacité de rafale de votre compte, l'API Athena émet une erreur similaire à la suivante : « » ClientError : Une erreur s'est produite (ThrottlingException) lors de l'appel de l'opération : Dépassement du débit<API_name>. » Réduisez le nombre d'appels par seconde ou la capacité de transmission en mode rafale pour l'API pour ce compte.

Le quota Athena pour les appels d'API par compte ne peut pas être modifié dans la console Athena Service Quotas. Pour demander une augmentation du quota pour les appels d'API Athena, accédez à la page Augmentation des limites de service AWS Support, puis complétez et envoyez le formulaire.

Gestion des versions du moteur Athena

Athena publie occasionnellement une nouvelle version du moteur pour améliorer les performances, les fonctionnalités et les corrections de code. Lorsqu'une nouvelle version du moteur est disponible, Athena vous en informe via la console Athena et votre AWS Health Dashboard. Vous êtes AWS Health Dashboard informé des événements susceptibles d'affecter vos AWS services ou votre compte. Pour plus d'informations sur AWS Health Dashboard, consultez Getting started with the AWS Health Dashboard.

La gestion des versions du moteur est configurée par groupe de travail. Vous pouvez utiliser des groupes de travail pour contrôler le moteur de requêtes utilisé par vos requêtes et pour savoir si Athena doit automatiquement mettre à niveau vos groupes de travail. Le moteur de requête utilisé est affiché dans l'éditeur de requêtes, sur la page des détails du groupe de travail, et est disponible via les API Athena.

- Par défaut, les groupes de travail sont configurés pour une mise à niveau automatique. Lorsqu'un groupe de travail est configuré pour une mise à niveau automatique, Athena le met à niveau pour vous, sauf si elle détecte des incompatibilités.
- Si vous configurez un groupe de travail pour utiliser une version donnée, Athena ne modifiera pas la version du groupe de travail.

Dans les deux cas, Athena met à niveau vos groupes de travail lorsqu'une version n'est plus disponible. Athena vous indique quand une version du moteur ne sera plus proposée. AWS Health Dashboard Vous êtes AWS Health Dashboard informé des événements susceptibles d'affecter vos AWS services ou votre compte. Pour plus d'informations sur AWS Health Dashboard, consultez Getting started with the AWS Health Dashboard.

Lorsque vous commencez à utiliser une nouvelle version du moteur, un petit sous-ensemble de requêtes peut être interrompu en raison d'incompatibilités. Les modifications majeures sont annoncées lors de la sortie d'une nouvelle version d'Athena. Vous devez utiliser des groupes de travail pour tester vos requêtes avant la mise à niveau en créant un groupe de travail de test qui utilise le nouveau moteur ou en testant la mise à niveau d'un groupe de travail existant. Pour plus d'informations, voir Essai des requêtes avant la mise à jour d'une version du moteur.

Rubriques

Modification des versions du moteur Athena

Athena publie occasionnellement une nouvelle version du moteur pour améliorer les performances, les fonctionnalités et les corrections de code. Lorsqu'une nouvelle version du moteur est disponible, Athena vous en informe dans la console. Vous pouvez choisir de laisser Athena décider du moment de la mise à jour ou spécifier manuellement une version du moteur Athena par groupe de travail.

Rubriques

Recherche de la version du moteur de requête pour un groupe de travail

Vous pouvez utiliser la page Workgroups (Groupes de travail) pour trouver la version actuelle du moteur pour tout groupe de travail.

Recherche de la version actuelle du moteur pour tout groupe de travail

1. Ouvrez la console Athena à l’adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le panneau de navigation de la console Athena, choisissez Workgroups (Groupes de travail).
4. Sur la page Workgroups (Groupes de travail), trouvez le groupe de travail souhaité. La colonne Query engine version (Version du moteur de requête) pour le groupe de travail affiche la version du moteur de requête.

Modification de la version du moteur dans la console Athena

Lorsqu'une nouvelle version du moteur est disponible, vous pouvez choisir de laisser Athena décider de la mise à jour du groupe de travail ou spécifier manuellement la version du moteur Athena utilisée par le groupe de travail. Si une seule version est actuellement disponible, il n'est pas possible de spécifier manuellement une version différente.

> **Note**
>
> Pour modifier la version du moteur d'un groupe de travail, vous devez avoir l'autorisation d'exécuter l'action `athena:ListEngineVersions` sur le groupe de travail. Pour des exemples de politiques IAM, voir Exemples de politiques de groupe de travail.

Laisser Athena décider de la mise à jour du groupe de travail

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.

2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Dans le panneau de navigation de la console, choisissez Workgroups (Groupes de travail).
4. Dans la liste des groupes de travail, choisissez le lien pour le groupe de travail que vous voulez configurer.
5. Choisissez Edit (Modifier).
6. Sous Query engine version (Version du moteur de requête), pour Update query engine (Mettre à jour le moteur de requête), choisissez Automatic (Automatique) pour laisser Athena décider quand mettre à jour de votre groupe de travail. Il s’agit du paramètre par défaut.
7. Sélectionnez Enregistrer les modifications.

   Dans la liste des groupes de travail, l'état de mise à jour du moteur de requête pour le groupe de travail indique Automatic (Automatique).

Choisir manuellement une version du moteur

1. Ouvrez la console Athena à l’adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Dans le panneau de navigation de la console, choisissez Workgroups (Groupes de travail).
4. Dans la liste des groupes de travail, choisissez le lien pour le groupe de travail que vous voulez configurer.
5. Choisissez Edit (Modifier).
6. Dans la section Query engine version (Version du moteur de requête), pour Update query engine (Mise à jour du moteur de requête), choisissez Manual (Manuel) pour choisir manuellement une version du moteur.
7. Utilisez l'option Query engine version (Version du moteur de requête) pour choisir la version du moteur à utiliser par le groupe de travail. Si aucune version du moteur différente n'est disponible, une version du moteur différente ne peut pas être spécifiée.
8. Sélectionnez Enregistrer les modifications.

   Dans la liste des groupes de travail, l'état de mise à jour du moteur de requête pour le groupe de travail indique Manual (Manuel).

Modification de la version du moteur à l'aide du AWS CLI

Pour modifier la version du moteur à l'aide de AWS CLI, utilisez la syntaxe de l'exemple suivant.

```
aws athena update-work-group --work-group workgroup-name --configuration-updates
 EngineVersion={SelectedEngineVersion='Athena engine version 3'}
```

Spécification de la version du moteur lors de la création d'un groupe de travail

Lorsque vous créez un groupe de travail, vous pouvez spécifier la version du moteur que le groupe de travail utilise ou laisser Athena décider de la mise à jour du groupe de travail. Si une nouvelle version du moteur est disponible, une bonne pratique consiste à créer un groupe de travail pour tester le nouveau moteur avant de mettre à jour vos autres groupes de travail. Pour spécifier la version du moteur d'un groupe de travail, vous devez avoir l'autorisation `athena:ListEngineVersions` sur le groupe de travail. Pour des exemples de politiques IAM, voir Exemples de politiques de groupe de travail.

Spécifier la version du moteur lors de la création d'un groupe de travail

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Dans le panneau de navigation de la console, choisissez Workgroups (Groupes de travail).
4. Sur la page Workgroups (Groupes de travail), choisissez Create workgroup (Créer un groupe de travail).
5. Sur la page Create workgroup (Créer un groupe de travail), dans la section Query engine version (Version du moteur de requête), effectuez l'une des opérations suivantes :
   - Choisissez Automatic (Automatique) pour laisser Athena décider quand mettre à jour de votre groupe de travail. Il s'agit du paramètre par défaut.
   - Choisissez Manual (Manuel) pour choisir manuellement une version de moteur différente si elle est disponible.
6. Saisissez les informations pour les autres champs si nécessaire. Pour plus d'informations sur les autres champs, voir Créer un groupe de travail.
7. Choisissez Create workgroup (Créer un groupe de travail).

Essai des requêtes avant la mise à jour d'une version du moteur

Lorsqu'un groupe de travail est mis à jour vers une nouvelle version du moteur, certaines de vos requêtes peuvent être interrompues en raison d'incompatibilités. Pour vous assurer que la mise à jour de votre moteur se déroule sans problème, vous pouvez tester vos requêtes à l'avance.

Essai de vos requêtes avant une mise à jour de la version du moteur

1. Vérifiez la version du moteur du groupe de travail que vous utilisez. La version du moteur que vous utilisez est affichée sur la page Workgroups (Groupes de travail) dans la colonne Query engine version (Version du moteur de requête) pour le groupe de travail. Pour plus d'informations, consultez Recherche de la version du moteur de requête pour un groupe de travail.
2. Créez un groupe de travail de test qui utilise la nouvelle version du moteur. Pour plus d'informations, consultez Spécification de la version du moteur lors de la création d'un groupe de travail.
3. Utilisez le nouveau groupe de travail pour exécuter les requêtes que vous voulez tester.
4. Si une requête échoue, utilisez la version Référence de la version du moteur Athena pour vérifier les modifications qui pourraient affecter la requête. Certaines modifications peuvent nécessiter la mise à jour de la syntaxe de vos requêtes.
5. Si vos requêtes échouent toujours, contactez AWS Support pour obtenir de l'aide. Dans AWS Management Console, choisissez Support, Support Center (Centre de support), ou posez une question sur AWS re:Post en utilisant l'étiquette Amazon Athena.

Dépannage des requêtes qui échouent

Si une requête échoue après une mise à jour de la version du moteur, utilisez la version Référence de la version du moteur Athena pour vérifier s'il y a des interruptions ou d'autres modifications, y compris celles pouvant affecter la syntaxe de vos requêtes.

Si vos requêtes échouent toujours, contactez AWS Support pour obtenir de l'aide. Dans le AWS Management Console, choisissez Support, Support Center ou posez une question sur AWS Re:post en utilisant le tag Amazon Athena.

Référence de la version du moteur Athena

Cette section répertorie les modifications apportées au moteur de requête Athena.

Rubriques

Version 3 du moteur Athena

Pour la version 3 du moteur, Athena a introduit une approche d'intégration continue de la gestion des logiciels open source. Celle-ci améliore la simultanéité des projets Trino et Presto, afin que vous puissiez accéder plus rapidement aux améliorations de la communauté, intégrées et ajustées au sein du moteur Athena.

Cette version 3 du moteur Athena prend en charge toutes les fonctionnalités de la version 2. Ce document met en évidence les principales différences entre les versions 2 et 3 du moteur Athena. Pour plus d'informations, consultez l'article du blog AWS Big Data sur la mise à niveau vers la version 3 du moteur Athena pour améliorer les performances des requêtes et accéder à davantage de fonctionnalités d'analyse.

Premiers pas

Pour commencer, créez un nouveau groupe de travail Athena utilisant la version 3 du moteur Athena ou configurez un groupe de travail existant pour qu'il utilise la version 3. Tout groupe de travail

Athena peut effectuer une mise à niveau de la version 2 vers la version 3 du moteur sans interruption de votre capacité à soumettre des requêtes.

Pour plus d'informations, consultez la rubrique Modification des versions du moteur Athena.

Améliorations et nouvelles fonctions

Les fonctionnalités et mises à jour répertoriées incluent des améliorations provenant d'Athena lui-même et de fonctionnalités intégrées à partir de Trino open source. Pour consulter une liste exhaustive des opérateurs et des fonctions de requête SQL, consultez la documentation de Trino.

Fonctionnalités ajoutées

Prise en charge des algorithmes de mise en compartiments Apache Spark

Athena peut lire les compartiments générés par l'algorithme de hachage Spark. Pour spécifier que les données ont été initialement rédigées par l'algorithme de hachage Spark, insérez `('bucketing_format'='spark')` dans la clause `TBLPROPERTIES` de votre instruction `CREATE TABLE`. Si cette propriété n'est pas spécifiée, l'algorithme de hachage Hive est utilisé.

```
CREATE EXTERNAL TABLE `spark_bucket_table`(
  `id` int,
  `name` string
  )
CLUSTERED BY (`name`)
INTO 8 BUCKETS
STORED AS PARQUET
LOCATION
  's3://DOC-EXAMPLE-BUCKET/to/bucketed/table/'
TBLPROPERTIES ('bucketing_format'='spark')
```

Fonctions ajoutées

Les fonctions de cette section sont nouvelles pour la version 3 du moteur Athena.

Fonctions d'agrégation

listagg(x, separator) : renvoie les valeurs d'entrée concaténées, séparées par la chaîne de séparation.

```
SELECT listagg(value, ',') WITHIN GROUP (ORDER BY value) csv_value
FROM (VALUES 'a', 'c', 'b') t(value);
```

Fonctions de tableau

contains_sequence(x, seq) : renvoie « true » (vrai) si le tableau x contient l'ensemble du tableau seq sous forme de sous-ensemble séquentiel (toutes les valeurs dans le même ordre consécutif).

```
SELECT contains_sequence(ARRAY [1,2,3,4,5,6], ARRAY[1,2]);
```

Fonctions binaires

murmur3 (binary) — Calcule le hachage de 128 bits MurmurHash sur 3 du binaire.

```
SELECT murmur3(from_base64('aaaaaa'));
```

Fonctions de conversion

format_number(number) : renvoie une chaîne formatée à l'aide d'un symbole d'unité.

```
SELECT format_number(123456); -- '123K'
```

```
SELECT format_number(1000000); -- '1M'
```

Fonctions de date et d’heure

timezone_hour(timestamp) : renvoie l'heure du décalage de fuseau horaire par rapport à l'horodatage.

```
SELECT EXTRACT(TIMEZONE_HOUR FROM TIMESTAMP '2020-05-10 12:34:56 +08:35');
```

timezone_minute(timestamp) : renvoie la minute du décalage de fuseau horaire par rapport à l'horodatage.

```
SELECT EXTRACT(TIMEZONE_MINUTE FROM TIMESTAMP '2020-05-10 12:34:56 +08:35');
```

Fonctions géospatiales

to_encoded_polyline(Geometry) : encode une linestring ou un multipoint en polyligne.

```
SELECT to_encoded_polyline(ST_GeometryFromText(
   'LINESTRING (-120.2 38.5, -120.95 40.7, -126.453 43.252)'));
```

from_encoded_polyline(varchar) : décode un polyligne en linestring.

```
SELECT ST_AsText(from_encoded_polyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@'));
```

to_geojson_geometry (SphericalGeography) — Renvoie la géographie sphérique spécifiée au format GeoJSON.

```
SELECT to_geojson_geometry(to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (0 0, 1 2, 3 4)')));
```

from_geojson_geometry(varchar) : renvoie l'objet de type géographique sphérique à partir de la représentation GeoJSON, en supprimant les clés/valeurs non géométriques. `Feature` et `FeatureCollection` ne sont pas pris en charge.

```
SELECT
 from_geojson_geometry(to_geojson_geometry(to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (0 0, 1 2, 3 4)'))));
```

geometry_nearest_points(Geometry, Geometry) : renvoie les points les plus proches les uns des autres sur chaque géométrie. Si l'une des géométries est vide, renvoie NULL (nul). Dans le cas contraire, renvoie une ligne de deux objets `Point` ayant la distance minimale de deux points quelconques sur les géométries. Le premier point provient du premier argument de géométrie, le second du second argument de géométrie. S'il existe plusieurs paires ayant la même distance minimale, une paire est choisie arbitrairement.

```
SELECT geometry_nearest_points(ST_GeometryFromText(
   'LINESTRING (50 100, 50 200)'), ST_GeometryFromText(
   'LINESTRING (10 10, 20 20)'));
```

Fonctions Set Digest

make_set_digest(x) : compose toutes les valeurs d'entrée de x dans un setdigest.

```
SELECT make_set_digest(value) FROM (VALUES 1, 2, 3) T(value);
```

Fonctions de chaîne

soundex (char) : renvoie une chaîne de caractères contenant la représentation phonétique de char.

```
SELECT name
```

```
FROM nation
WHERE SOUNDEX(name) = SOUNDEX('CHYNA'); -- CHINA
```

concat_ws(string0, string1, …, stringN) : renvoie la concaténation de `string1, string2, ..., stringN` avec `string0` comme séparateur. Si `string0` a la valeur NULL, la valeur de retour est NULL. Toutes les valeurs nulles fournies dans les arguments après le séparateur sont ignorées.

```
SELECT concat_ws(',', 'def', 'pqr', 'mno');
```

Fonctions de fenêtrage

GROUPS : intègre une prise en charge des cadres de fenêtre basés sur des groupes.

```
SELECT array_agg(a) OVER(
   ORDER BY a ASC NULLS FIRST GROUPS BETWEEN 1 PRECEDING AND 2 FOLLOWING)
FROM (VALUES 3, 3, 3, 2, 2, 1, null, null) T(a);
```

Améliorations des performances

La version 3 du moteur Athena comprend les améliorations de performances suivantes.

- Récupération plus rapide des métadonnées des AWS Glue tables : les requêtes impliquant plusieurs tables réduiront le temps de planification des requêtes.
- Filtrage dynamique pour les JOINTURES DROITES : le filtrage dynamique est désormais activé pour les jointures droites qui ont des conditions de jointure égales, comme dans l'exemple suivant.

  ```
  SELECT *
  FROM lineitem RIGHT JOIN tpch.tiny.supplier
  ON lineitem.suppkey = supplier.suppkey
  WHERE supplier.name = 'abc';
  ```

- Instructions préparées volumineuses – Augmentation de la taille par défaut de l'en-tête de demande/réponse HTTP à 2 Mo pour permettre les instructions préparées volumineuses.
- approx_percentile() – La fonction `approx_percentile` utilise désormais `tdigest` au lieu de `qdigest` pour récupérer des quantiles approximatives à partir des distributions. Cela permet d'améliorer les performances et de réduire l'utilisation de la mémoire. Notez qu'à la suite de cette modification, la fonction renvoie des résultats différents de ceux qu'elle renvoyait dans la version 2 du moteur Athena. Pour plus d'informations, consultez La fonction approx_percentile renvoie des résultats différents.

Améliorations de la fiabilité

L'utilisation générale de la mémoire du moteur et le suivi dans la version 3 du moteur Athena ont été améliorés. Les requêtes volumineuses sont moins susceptibles d'échouer en cas de panne de nœud.

Améliorations de la syntaxe des requêtes

INTERSECT ALL : ajout de la prise en charge de `INTERSECT ALL`.

```
SELECT * FROM (VALUES 1, 2, 3, 4) INTERSECT ALL SELECT * FROM (VALUES 3, 4);
```

EXCEPT ALL : ajout de la prise en charge de `EXCEPT ALL`.

```
SELECT * FROM (VALUES 1, 2, 3, 4) EXCEPT ALL SELECT * FROM (VALUES 3, 4);
```

RANGE PRECEDING : ajout de la prise en charge de `RANGE PRECEDING` dans les fonctions de fenêtrage.

```
SELECT sum(x) over (order by x range 1 preceding)
FROM (values (1), (1), (2), (2)) t(x);
```

MATCH_RECOGNIZE : ajout de la prise en charge de la mise en correspondance des modèles de lignes, comme dans l'exemple suivant.

```
SELECT m.id AS row_id, m.match, m.val, m.label
FROM (VALUES(1, 90),(2, 80),(3, 70),(4, 70)) t(id, value)
MATCH_RECOGNIZE (
        ORDER BY id
        MEASURES match_number() AS match,
        RUNNING LAST(value) AS val,
        classifier() AS label
        ALL ROWS PER MATCH
        AFTER MATCH SKIP PAST LAST ROW
        PATTERN (() | A) DEFINE A AS true
) AS m;
```

Améliorations du format et du type de données

La version 3 du moteur Athena comporte les améliorations suivantes en matière de format et de type de données.

- LZ4 et ZSTD : ajout de la prise en charge de la lecture des données Parquet compressées LZ4 et ZSTD. Ajout de la prise en charge de l'écriture de données ORC compressées ZSTD.
- Tables basées sur des liens symboliques : ajout de la prise en charge de la création de tables basées sur des liens symboliques sur les fichiers Avro. Un exemple suit.

```
CREATE TABLE test_avro_symlink
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
...
INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
```

- SphericalGeography— Le SphericalGeography type fournit un support natif pour les entités spatiales représentées sur les coordonnées géographiques (parfois appelées coordonnées géodésiques, lat/lon ou lon/lat). Les coordonnées géographiques sont des coordonnées sphériques exprimées en unités angulaires (degrés).

  La fonction `to_spherical_geography` renvoie des coordonnées géographiques (sphériques) à partir de coordonnées géométriques (planes), comme dans l'exemple suivant.

```
SELECT to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (-40.2 28.9, -40.2 31.9, -37.2 31.9)'));
```

Évolutions

Lorsque vous migrez de la version 2 vers la version 3 du moteur Athena, certaines modifications peuvent affecter le schéma des tables, la syntaxe ou l'utilisation des types de données. Cette section répertorie les messages d'erreur associés et propose des solutions de contournement.

Modifications de la syntaxe de requête

IGNORE NULLS ne peut pas être utilisé avec des fonctions de fenêtrage sans valeur

Message d'erreur : Impossible de spécifier une clause de traitement nulle pour la fonction `bool_or`.

Cause : `IGNORE NULLS` ne peut désormais être utilisé qu'avec les fonctions de valeur `first_value`, `last_value`, `nth_value`, `lead` et `lag`. Cette modification a été apportée pour se conformer à la spécification SQL ANSI.

Solution suggérée : Supprimer `IGNORE NULLS` des fonctions de fenêtrage sans valeur dans les chaînes de requête.

La fonction CONCAT doit comporter deux arguments ou plus.

Message d'erreur : INVALID_FUNCTION_ARGUMENT: There must be two or more concatenation arguments (INVALID_FUNCTION_ARGUMENT : il doit y avoir au moins deux arguments de concaténation)

Cause : auparavant, la fonction de chaîne `CONCAT` acceptait un seul argument. Dans la version 3 du moteur Athena, la fonction `CONCAT` nécessite un minimum de deux arguments.

Solution suggérée : remplacez les occurrences `CONCAT(str)` par `CONCAT(str, '')`.

Dans la version 3 du moteur Athena, les fonctions ne peuvent pas comporter plus de 127 arguments. Pour plus d'informations, consultez Trop d'arguments pour un appel de fonction.

La fonction approx_percentile renvoie des résultats différents

La fonction `approx_percentile` renvoie dans la version 3 du moteur Athena des résultats différents de ceux de la version 2 du moteur Athena.

Messages d'erreur : aucun.

Cause : la fonction `approx_percentile` est sujette à des modifications de version.

> **Important**
>
> Comme les sorties de la fonction `approx_percentile` sont des approximations et que les approximations sont susceptibles de changer d'une version à l'autre, vous ne devez pas vous fier à la fonction `approx_percentile` pour les applications critiques.

Solution suggérée : pour se rapprocher du comportement de la version 2 du moteur Athena de la fonction `approx_percentile`, vous pouvez utiliser un ensemble de fonctions différent dans la version 3 du moteur Athena. Supposons, par exemple, que vous ayez la requête suivante dans la version 2 du moteur Athena :

```
SELECT approx_percentile(somecol, 2E-1)
```

Pour obtenir approximativement le même résultat dans la version 3 du moteur Athena, vous pouvez essayer les fonctions `qdigest_agg` et `value_at_quantile`, comme dans l'exemple suivant. Notez que, même avec cette solution de contournement, le même comportement n'est pas garanti.

```
SELECT value_at_quantile(qdigest_agg(somecol, 1), 2E-1)
```

La fonction géospatiale ne prend pas en charge l'entrée varbinary

Message d'erreur : FUNCTION_NOT_FOUND for st_XXX (FUNCTION_NOT_FOUND pour st_XXX)

Cause : certaines fonctions géospatiales ne prennent plus en charge le type d'entrée `VARBINARY` hérité ou les signatures de fonctions liées au texte.

Solution suggérée : utilisez les fonctions géospatiales pour convertir les types d'entrée en types pris en charge. Les types d'entrée pris en charge sont indiqués dans le message d'erreur.

Dans les clauses GROUP BY, les colonnes imbriquées doivent être entre guillemets doubles

Message d'erreur : « *`column_name`* ». » « *`nested_column`* » doit être une expression agrégée ou apparaître dans la clause GROUP BY

Cause : la version 3 du moteur Athena exige que les noms de colonnes imbriqués dans les clauses `GROUP BY` soient placés entre guillemets doubles. Par exemple, la requête suivante produit l'erreur car, dans la clause `GROUP BY`, `user.name` n'est pas entre guillemets doubles.

```
SELECT "user"."name" FROM dataset
GROUP BY user.name
```

Solution suggérée : placez des guillemets autour des noms de colonnes imbriqués dans les clauses `GROUP BY`, comme dans l'exemple suivant.

```
SELECT "user"."name" FROM dataset
GROUP BY "user"."name"
```

FilterNode Erreur inattendue lors de l'utilisation d'OPTIMIZE sur une table Iceberg

Message d'erreur : Inattendu FilterNode détecté dans le plan ; le connecteur n'a probablement pas pu gérer l'expression WHERE fournie.

Cause : L'`OPTIMIZE`instruction exécutée sur la table Iceberg utilisait une `WHERE` clause qui incluait une colonne non partitionnée dans son expression de filtre.

Solution suggérée : L'`OPTIMIZE`instruction prend en charge le filtrage par partitions uniquement. Lorsque vous exécutez `OPTIMIZE` sur des tables partitionnées, incluez uniquement les colonnes

de partition dans la `WHERE` clause. Si vous exécutez `OPTIMIZE` sur une table non partitionnée, ne spécifiez aucune clause. `WHERE`

Ordre des arguments de la fonction Log()

Dans la version 2 du moteur Athena, l'ordre des arguments de la fonction `log()` était `log(value, base)`. Dans la version 3 du moteur Athena, il a été modifié en `log(base, value)`, conformément aux normes SQL.

La fonction Minute() ne prend pas en charge le type de données « interval year to month » (intervalle de l'année au mois)

Message d'erreur : Unexpected parameters (interval year to month) for function minute. (Paramètres inattendus [intervalle année-mois] pour la fonction minute.) Prévu : minute(timestamp with time zone) [minute(horodatage avec fuseau horaire)], minute(time with time zone) [minute(heure avec fuseau horaire)], minute(timestamp) [minute(horodatage)], minute(time) [minute(heure)], minute(interval day to second) [minute(intervalle d'une journée à une seconde)].

Cause : dans la version 3 du moteur Athena, les vérifications de type ont été rendues plus précises pour `EXTRACT`, conformément à la spécification SQL ANSI.

Solution suggérée : mettez à jour les requêtes pour vous assurer que les types correspondent aux signatures de fonctions suggérées.

Les expressions ORDER BY doivent apparaître dans la liste SELECT

Message d'erreur : For SELECT DISTINCT, ORDER BY expressions must appear in SELECT list (Pour SELECT DISTINCT, les expressions ORDER BY doivent apparaître dans la liste SELECT)

Cause : un crénelage de table incorrect est utilisé dans une clause `SELECT`.

Solution suggérée : vérifiez que toutes les colonnes de l'expression `ORDER BY` ont des références appropriées dans la clause `SELECT DISTINCT`.

Échec de la requête lors de la comparaison de plusieurs colonnes renvoyées par une sous-requête

Exemple de message d'erreur : l'expression de valeur et le résultat de la sous-requête doivent être du même type : ligne (varchar, varchar) et ligne (ligne (varchar, varchar))

Cause : en raison d'une mise à jour de syntaxe dans la version 3 du moteur Athena, cette erreur se produit lorsqu'une requête tente de comparer plusieurs valeurs renvoyées par une sous-requête et

que l'instruction SELECT de sous-requête met sa liste de colonnes entre parenthèses, comme dans l'exemple suivant.

```
SELECT *
FROM table1
WHERE (t1_col1, t1_col2)
IN (SELECT (t2_col1, t2_col2) FROM table2)
```

Solution : dans la version 3 du moteur Athena, supprimez les parenthèses entourant la liste des colonnes dans l'instruction SELECT de sous-requête, comme dans l'exemple de requête mis à jour suivant.

```
SELECT *
FROM table1
WHERE (t1_col1, t1_col2)
IN (SELECT t2_col1, t2_col2 FROM table2)
```

SKIP est un mot réservé pour les requêtes DML.

Le mot SKIP est désormais un mot réservé aux requêtes DML telles que SELECT. Pour utiliser SKIP en tant qu'identifiant dans une requête DML, mettez-le entre guillemets.

Pour plus d'informations sur les mots réservés dans Athena, consultez Mots-clés réservés.

Clauses SYSTEM_TIME et SYSTEM_VERSION obsolètes pour les voyages dans le temps

Message d'erreur : mismatched input 'SYSTEM_TIME'. (entrée « SYSTEM_TIME » non concordante.) En attente de : « TIMESTAMP », « VERSION »

Cause : dans la version 2 du moteur Athena, les tables Iceberg utilisaient les clauses FOR SYSTEM_TIME AS OF et FOR SYSTEM_VERSION AS OF pour l'horodatage et le voyage dans le temps des versions. La version 3 du moteur Athena utilise les clauses FOR TIMESTAMP AS OF et FOR VERSION AS OF.

Solution suggérée : mettez à jour la requête SQL afin d'utiliser les clauses TIMESTAMP AS OF et VERSION AS OF pour les opérations de voyage dans le temps, comme dans les exemples suivants.

Voyage dans le temps par horodatage :

```
SELECT * FROM TABLE FOR TIMESTAMP AS OF (current_timestamp - interval '1' day)
```

Voyage dans le temps par version :

```
SELECT * FROM TABLE FOR VERSION AS OF 949530903748831860
```

Trop d'arguments pour un constructeur de tableaux

Message d'erreur : TOO_MANY_ARGUMENTS : trop d'arguments pour le constructeur de tableau.

Cause : le nombre maximum d'éléments dans un constructeur de tableau est désormais fixé à 254.

Solution suggérée : divisez les éléments en plusieurs tableaux de 254 éléments ou moins chacun, et utilisez la fonction `CONCAT` pour concaténer les tableaux, comme dans l'exemple suivant.

```
CONCAT(
ARRAY[x1,x2,x3...x254],
ARRAY[y1,y2,y3...y254],
...
)
```

L'identifiant délimité par une longueur nulle n'est pas autorisé

Message d'erreur : Zero-length delimited identifier not allowed. (L'identifiant délimité par une longueur nulle n'est pas autorisé.)

Cause : une requête utilisait une chaîne vide comme alias de colonne.

Solution suggérée : mettez à jour la requête afin d'utiliser un alias non vide pour la colonne.

Modifications du traitement des données

Validation du compartiment

Message d'erreur : HIVE_INVALID_BUCKET_FILES : la table Hive est corrompue..

Cause : la table a peut-être été corrompue. Pour garantir l'exactitude des requêtes des tables compartimentées, la version 3 du moteur Athena permet une validation supplémentaire sur les tables compartimentées afin de garantir l'exactitude des requêtes et d'éviter des échecs inattendus lors de l'exécution.

Solution suggérée : recréez la table à l'aide de la version 3 du moteur Athena.

La conversion d'un struct au format JSON renvoie désormais des noms de champs.

Lorsque vous convertissez un `struct` au format JSON dans une requête `SELECT` dans la version 3 du moteur Athena, la conversion renvoie désormais à la fois les noms des champs et les valeurs (par exemple, « `useragent":null` »), plutôt qu'uniquement les valeurs (par exemple, `null`).

Modification de l'application de la sécurité au niveau des colonnes de la table Iceberg

Message d'erreur : Access Denied: Cannot select from columns (Accès refusé : impossible de sélectionner parmi les colonnes)

Cause : la table Iceberg a été créée en dehors d'Athena et utilise une version du kit SDK Apache Iceberg antérieure à 0.13.0. Comme les versions antérieures du SDK ne remplissaient pas de colonnes AWS Glue, Lake Formation n'a pas pu déterminer les colonnes autorisées à accéder.

Solution suggérée : effectuez une mise à jour à l'aide de l'instruction Athena ALTER TABLE SET PROPERTIES ou utilisez la dernière version du kit SDK Iceberg pour corriger la table et mettre à jour les informations des colonnes dans AWS Glue.

Les valeurs nulles dans les types de données List sont désormais propagées aux fonctions définies par l'utilisateur (UDF)

Message d'erreur : Null Pointer Exception (Exception de pointeur nul)

Cause : ce problème peut vous affecter si vous utilisez le connecteur UDF et avez implémenté une fonction Lambda définie par l'utilisateur.

La version 2 du moteur Athena a filtré les valeurs nulles dans les types de données List transmis à une fonction définie par l'utilisateur. Dans la version 3 du moteur Athena, les valeurs nulles sont désormais préservées et transmises à l'UDF. Cela peut provoquer une exception de pointeur nul si l'UDF tente de déréférencer l'élément nul sans vérification.

Par exemple, si les données `[null, 1, null, 2, 3, 4]` se trouvent dans une source de données d'origine telle que DynamoDB, les éléments suivants sont transmis à la fonction Lambda définie par l'utilisateur :

Version 2 du moteur Athena : `[1,2,3,4]`

Version 3 du moteur Athena : `[null, 1, null, 2, 3, 4]`

Solution suggérée : assurez-vous que votre fonction Lambda définie par l'utilisateur gère les éléments nuls dans les types de données List.

Les sous-chaînes des tableaux de caractères ne contiennent plus d'espaces rembourrés

Message d'erreur : No error is thrown, but the string returned no longer contains padded spaces. (Il n'y a pas d'erreur, mais la chaîne de caractères renvoyée ne contient plus d'espaces rembourrés.) Par exemple, `substr(char[20],1,100)` renvoie désormais une chaîne de longueur 20 au lieu de 100.

Solution suggérée : aucune action n'est requise.

Forçage d'un type de colonne décimale non pris en charge

Messages d'erreur *: HIVE_CURSOR_ERROR : Impossible de lire le fichier Parquet : s3://DOC-EXAMPLE-BUCKET/ path/file_name .parquet ou type de colonne non pris en charge (varchar) pour la colonne Parquet ([column_name]*

Cause : la version 2 du moteur Athena réussissait parfois (mais échouait fréquemment) lors de tentatives de forçage du type de données de `varchar` à la décimale. Comme la version 3 du moteur Athena comporte une validation du type qui vérifie que le type est compatible avant d'essayer de lire la valeur, ces tentatives de forçage échouent désormais toujours.

Solution suggérée : Pour la version 2 du moteur Athena et la version 3 du moteur Athena, modifiez votre schéma AWS Glue pour utiliser un type de données numérique plutôt que `varchar` pour les colonnes décimales dans les fichiers Parquet. Explorez à nouveau les données et assurez-vous que le nouveau type de données de colonne est de type décimal, ou recréez manuellement la table dans Athena et utilisez la syntaxe `decimal(`*`precision`*`, `*`scale`*`)` pour spécifier un type de données decimal pour la colonne.

Les valeurs NaN flottantes ou doubles ne peuvent plus être converties en bigint

Message d'erreur : INVALID_CAST_ARGUMENT : Impossible de convertir NaN réel/double en bigint

Cause : dans la version 3 du moteur Athena, `NaN` ne peut plus être converti en 0 en tant que `bigint`.

Solution suggérée : assurez-vous que les valeurs `NaN` ne sont pas présentes dans les colonnes `float` ou `double` lorsque vous convertissez en `bigint`.

changement de type de retour de la fonction uuid()

Le problème suivant concerne à la fois les tables et les vues.

Message d'erreur : Type Hive non pris en charge : uuid

Cause : dans la version 2 du moteur Athena, la fonction `uuid()` renvoyait une chaîne, mais dans la version 3 du moteur Athena, elle renvoie un pseudo UUID généré aléatoirement (type 4). Le type de données de colonne UUID n'étant pas pris en charge dans Athena, la fonction `uuid()` ne peut plus être utilisée directement dans les requêtes CTAS pour générer des colonnes UUID dans la version 3 du moteur Athena.

Par exemple, l'instruction `CREATE TABLE` suivante s'exécute correctement dans la version 2 du moteur Athena, mais renvoie NOT_SUPPORTED : Type Hive non pris en charge : uuid dans la version 3 du moteur Athena :

```
CREATE TABLE uuid_table AS
   SELECT uuid() AS myuuid
```

De même, l'instruction `CREATE VIEW` suivante s'exécute correctement dans la version 2 du moteur Athena, mais renvoie Type de colonne non valide pour colonne myuuid : type Hive non pris en charge : uuid dans la version 3 du moteur Athena :

```
CREATE VIEW uuid_view AS
   SELECT uuid() AS myuuid
```

Lorsqu'une vue ainsi créée dans la version 2 du moteur Athena est interrogée dans la version 3 du moteur Athena, une erreur semblable à la suivante se produit :

VIEW_IS_STALE : ligne 1:15 : La vue « awsdatacatalog.mydatabase.uuid_view » est obsolète ou dans un état non valide : la colonne [myuuid] de type uuid projetée depuis la vue de la requête à la position 0 ne peut pas être forcée vers la colonne [myuuid] de type varchar stockée dans la définition de la vue

Solution suggérée : lorsque vous créez la table ou la vue, utilisez la fonction `cast()` pour convertir la sortie de `uuid()` en un `varchar`, comme dans les exemples suivants :

```
CREATE TABLE uuid_table AS
   SELECT CAST(uuid() AS VARCHAR) AS myuuid
```

```
CREATE VIEW uuid_view AS
   SELECT CAST(uuid() AS VARCHAR) AS myuuid
```

Problèmes de forçage liés à CHAR et VARCHAR

Utilisez les solutions de contournement décrites dans cette section si vous rencontrez des problèmes de forçage avec `varchar` et `char` dans la version 3 du moteur Athena. Si vous ne parvenez pas à utiliser ces solutions de contournement, veuillez contacter. AWS Support

Échec de la fonction CONCAT avec des entrées CHAR et VARCHAR mixtes

Problème : la requête suivante réussit dans la version 2 du moteur Athena.

```
SELECT concat(CAST('abc' AS VARCHAR(20)), '12', CAST('a' AS CHAR(1)))
```

Cependant, dans la version 3 du moteur Athena, la même requête échoue avec ce qui suit :

Message d'erreur : FUNCTION_NOT_FOUND : ligne 1:8 : paramètres inattendus (varchar (20), varchar (2), char (1)) pour la fonction concat. Attendu : concat (char (x), char (y)), concat (tableau (E), E) E, concat (E, tableau (E)) E, concat (tableau (E)) E, concat (varchar), (varbinary)

Solution suggérée : lorsque vous utilisez la fonction `concat`, convertissez en `char` ou `varchar`, mais pas en un mélange des deux.

Échec de la concaténation SQL || avec les entrées CHAR et VARCHAR

Dans la version 3 du moteur Athena, l'opérateur de concaténation `||` à double barre verticale a besoin de `varchar` comme entrées. Les entrées ne peuvent pas être une combinaison de types `varchar` et `char`.

Message d'erreur : TYPE_NOT_FOUND : ligne 1:26 : type inconnu : char (65537)

Cause : une requête qui utilise `||` pour concaténer un `char` et un `varchar` peut produire l'erreur, comme dans l'exemple suivant.

```
SELECT CAST('a' AS CHAR) || CAST('b' AS VARCHAR)
```

Solution suggérée : concaténer `varchar` avec `varchar`, comme dans l'exemple suivant.

```
SELECT CAST('a' AS VARCHAR) || CAST('b' AS VARCHAR)
```

Échec des requêtes CHAR et VARCHAR UNION

Message d'erreur : NOT_SUPPORTED : type Hive non pris en charge : char (65536). Types de CHAR pris en charge : CHAR (<=255)

Cause : une requête qui tente de combiner `char` et `varchar`, comme dans l'exemple suivant :

```
CREATE TABLE t1 (c1) AS SELECT CAST('a' as CHAR) as c1 UNION ALL SELECT CAST('b' AS
 VARCHAR) AS c1
```

Solution suggérée : dans l'exemple de requête, convertissez `'a'` en `varchar` plutôt qu'en `char`.

Espaces vides indésirables après le forçage CHAR ou VARCHAR

Dans la version 3 du moteur Athena, lorsque les données `char(X)` et `varchar` sont forcées en un seul type lors de la formation d'un tableau ou d'une seule colonne, `char(65535)` est le type cible et chaque champ contient de nombreux espaces de fin indésirables.

Cause : la version 3 du moteur Athena force `varchar` et `char(X)` en `char(65535)`, puis remplit correctement les données avec des espaces.

Solution suggérée : convertissez chaque champ de manière explicite en `varchar`.

Modifications d'horodatage

Modification de comportement suite à la conversion d'un horodatage avec fuseau horaire en varchar

Dans la version 2 du moteur Athena, la conversion de `Timestamp` avec un fuseau horaire en `varchar` a provoqué la modification de certains littéraux du fuseau horaire (par exemple, `US/Eastern` est devenu `America/New_York`). Ce comportement ne survient pas dans la version 3 du moteur Athena.

Le dépassement de date et d'horodatage génère une erreur

Message d'erreur : Millis overflow: XXX (Dépassement millis : XXX)

Cause : Les dates ISO 8601 n'étant pas vérifiées dans la version 2 du moteur Athena, certaines dates produisent un horodatage négatif. La version 3 du moteur Athena vérifie la présence de ce dépassement et génère une exception.

Solution suggérée : assurez-vous que l'horodatage se situe dans la plage.

Fuseaux horaires politiques avec TIME non pris en charge

Message d'erreur : INVALID LITERAL

Cause : requêtes telles que `SELECT TIME '13:21:32.424 America/Los_Angeles'`.

Solution suggérée : évitez d'utiliser des fuseaux horaires politiques avec `TIME`.

Le décalage de précision dans les colonnes d'horodatage provoque une erreur de sérialisation

Message d'erreur : SERIALIZATION_ERROR: Could not serialize column '*COLUMNZ*' of type 'timestamp(3)' at position *X*:*Y* (SERIALIZATION_ERROR : impossible de sérialiser la colonne « COLUMNZ » de type « timestamp(3) » à la position X:Y)

COLUMNZ est le nom de sortie de la colonne à l'origine du problème. Les nombres *X*:*Y* indiquent la position de la colonne dans la sortie.

Cause : la version 3 du moteur Athena vérifie que la précision des horodatages des données est identique à la précision spécifiée pour le type de données de colonne dans la spécification de la table. Actuellement, cette précision est toujours de 3. Si les données ont une précision supérieure à cette valeur, les requêtes échouent et l'erreur est signalée.

Solution suggérée : vérifiez vos données pour vous assurer que vos horodatages ont une précision de l'ordre de la milliseconde.

Précision d'horodatage incorrecte dans les requêtes UNLOAD et CTAS pour les tables Iceberg

Message d'erreur : précision d'horodatage incorrecte pour horodatage (6) ; la précision configurée est de MILLISECONDES

Cause : la version 3 du moteur Athena vérifie que la précision des horodatages des données est identique à la précision spécifiée pour le type de données de colonne dans la spécification de la table. Actuellement, cette précision est toujours de 3. Si les données ont une précision supérieure à cette valeur (par exemple, microsecondes au lieu de millisecondes), les requêtes peuvent échouer et l'erreur est signalée.

Solution : pour contourner ce problème, commencez par `CAST` la précision de l'horodatage sur 6, comme dans l'exemple CTAS suivant qui crée une table Iceberg. Notez que la précision doit être spécifiée comme 6 au lieu de 3 pour éviter l'erreur Précision d'horodatage (3) non prise en charge pour Iceberg.

```
CREATE TABLE my_iceberg_ctas
WITH (table_type = 'ICEBERG', location = 's3://DOC-EXAMPLE-BUCKET/table_ctas/',
format = 'PARQUET')
AS SELECT id, CAST(dt AS timestamp(6)) AS "dt"
FROM my_iceberg
```

Ensuite, comme Athéna ne prend pas en charge l'horodatage 6, convertissez à nouveau la valeur en horodatage (par exemple, dans une vue). L'exemple suivant crée une vue à partir de la table `my_iceberg_ctas`.

```
CREATE OR REPLACE VIEW my_iceberg_ctas_view AS
SELECT cast(dt AS timestamp) AS dt
FROM my_iceberg_ctas
```

La lecture du type Long comme horodatage ou vice versa dans les fichiers ORC provoque désormais une erreur de fichier ORC mal formée

Message d'erreur : Error opening Hive split 'FILE (SPLIT POSITION)' Malformed ORC file. (Erreur lors de l'ouverture du fichier ORC mal formé « FILE (SPLIT POSITION) » du fractionnement Hive.) Impossible de lire l'horodatage de type SQL à partir du flux ORC .long_type de type LONG

Cause : la version 3 du moteur Athena rejette désormais tout forçage implicite du type de données `Long` en `Timestamp` ou de `Timestamp` en `Long`. Auparavant, les valeurs `Long` étaient implicitement converties en horodatage comme s'il s'agissait de millisecondes d'époque.

Solution suggérée : utilisez la fonction `from_unixtime` pour convertir explicitement la colonne, ou utilisez la fonction `from_unixtime` pour créer une colonne supplémentaire pour les requêtes futures.

Heure et intervalle d'une année au mois non pris en charge

Message d'erreur : TYPE MISMATCH

Cause : la version 3 du moteur Athena ne prend pas en charge l'heure et l'intervalle d'une année au mois (par exemple, `SELECT TIME '01:00' + INTERVAL '3' MONTH`).

Dépassement d'horodatage pour le format Parquet int96

Message d'erreur : timeOfDayNanos non valide

Cause : dépassement d'horodatage pour le format Parquet `int96`.

Solution suggérée : identifiez les fichiers spécifiques qui présentent ce problème. Générez ensuite à nouveau le fichier de données avec une up-to-date bibliothèque Parquet bien connue, ou utilisez Athena CTAS. Si le problème persiste, contactez le support d'Athena et indiquez-nous comment les fichiers de données sont générés.

Espace requis entre les valeurs de date et d'heure lors de la conversion d'une chaîne en un horodatage

Message d'erreur : INVALID_CAST_ARGUMENT : la valeur ne peut pas être convertie en horodatage.

Cause : la version 3 du moteur Athena n'accepte plus le tiret comme séparateur valide entre les valeurs de date et d'heure dans la chaîne d'entrée à `cast`. Par exemple, la requête suivante fonctionne dans la version 2 du moteur Athena, mais pas dans la version 3 du moteur Athena :

```
SELECT CAST('2021-06-06-23:38:46' AS timestamp) AS this_time
```

Solution suggérée : dans la version 3 du moteur Athena, remplacez le tiret entre la date et l'heure par un espace, comme dans l'exemple suivant.

```
SELECT CAST('2021-06-06 23:38:46' AS timestamp) AS this_time
```

changement de la valeur de retour de l'horodatage to_iso8601()

Message d'erreur : aucun.

Cause : dans la version 2 du moteur Athena, la fonction `to_iso8601` renvoie un horodatage avec le fuseau horaire même si la valeur transmise à la fonction n'inclut pas le fuseau horaire. Dans la version 3 du moteur Athena, la fonction `to_iso8601` renvoie un horodatage avec le fuseau horaire uniquement lorsque l'argument passé inclut le fuseau horaire.

Par exemple, la requête suivante transmet deux fois la date actuelle à la fonction `to_iso8601` : d'abord sous forme d'horodatage avec fuseau horaire, puis sous forme d'horodatage.

```
SELECT TO_ISO8601(CAST(CURRENT_DATE AS TIMESTAMP WITH TIME ZONE)),
 TO_ISO8601(CAST(CURRENT_DATE AS TIMESTAMP))
```

La sortie suivante montre le résultat de la requête dans chaque moteur.

Version 2 du moteur Athena :

#	_col0	_col1
1	2023-02-24T00:00:00.000Z	2023-02-24T00:00:00.000Z

Version 3 du moteur Athena :

#	_col0	_col1
1	2023-02-24T00:00:00.000Z	2023-02-24T00:00:00.000

Solution suggérée : pour reproduire le comportement précédent, vous pouvez transmettre la valeur d'horodatage à la fonction `with_timezone` avant de la transmettre à `to_iso8601`, comme dans l'exemple suivant :

```
SELECT to_iso8601(with_timezone(TIMESTAMP '2023-01-01 00:00:00.000', 'UTC'))
```

Résultat

#	_col0
1	2023-01-01T00:00:00.000Z

Le premier paramètre at_timezone() doit spécifier une date

Problème : dans la version 3 du moteur Athena, la fonction `at_timezone` ne peut pas prendre une valeur `time_with_timezone` comme premier paramètre.

Cause : sans informations de date, il est impossible de déterminer si la valeur transmise est l'heure d'été ou l'heure normale. Par exemple, `at_timezone('12:00:00 UTC', 'America/Los_Angeles')` est ambigu car il n'existe aucun moyen de déterminer si la valeur transmise est l'heure d'été du Pacifique (PDT) ou l'heure normale du Pacifique (PST).

Limites

La version 3 du moteur Athena présente les limitations suivantes.

- Performances des requêtes : de nombreuses requêtes s'exécutent plus rapidement sur la version 3 du moteur Athena, mais certains plans de requêtes peuvent différer de la version 2 du moteur Athena. Par conséquent, certaines requêtes peuvent différer en termes de latence ou de coût.
- Connecteurs Trino et Presto : les connecteurs Trino et Presto ne sont pas pris en charge. Utilisation d'une requête fédérée d'Amazon Athena pour vous connecter aux sources de données. Pour plus d'informations, consultez Utilisation de la requête fédérée d'Amazon Athena.

- Exécution tolérante aux pannes : l'exécution tolérante aux pannes de Trino (Trino Tardigrade) n'est pas prise en charge.
- Limite de paramètres de fonction – Les fonctions ne peuvent pas comporter plus de 127 paramètres. Pour plus d'informations, consultez Trop d'arguments pour un appel de fonction.

Les limites suivantes ont été introduites dans la version 2 du moteur Athena afin de s'assurer que les requêtes n'échouent pas en raison de limitations de ressources. Ces limites ne sont pas configurables par les utilisateurs.

- Nombre d'éléments de résultat – Le nombre d'éléments de résultat n est limité à 10 000 ou moins pour les fonctions suivantes : `min(col, n)`, `max(col, n)`, `min_by(col1, col2, n)` et `max_by(col1, col2, n)`.
- JEUX DE GROUPES – Le nombre maximal de tranches dans un jeu de groupes est 2048.
- Longueur maximale de la ligne du fichier texte : la longueur de ligne maximale par défaut pour les fichiers texte est de 200 Mo.
- Taille maximale des résultats de la fonction de séquence – La taille maximale de résultat d'une fonction de séquence est de 50 000 entrées. Par exemple, `SELECT sequence(0,45000,1)` réussit, mais `SELECT sequence(0,55000,1)` échoue avec le message d'erreur Le résultat de la fonction de séquence ne doit pas comporter plus de 50 000 entrées. Cette limite s'applique à tous les types d'entrées pour les fonctions de séquence, y compris aux horodatages.

Version 2 du moteur Athena

La version 2 du moteur Athena apporte les modifications suivantes.

- Améliorations et nouvelles fonctions
  - Amélioration des regroupements, des jointures et des sous-requêtes
  - Amélioration des types de données
  - Fonctions ajoutées
  - Améliorations des performances
  - Améliorations relatives à JSON
- Évolutions
  - Correctifs de bogue
  - Modifications apportées aux fonctions géospatiales

- Conformité ANSI SQL
- Fonctions remplacées
- Limites

Améliorations et nouvelles fonctions

- EXPLAIN et EXPLAIN ANALYZE – Vous pouvez utiliser l'instruction `EXPLAIN` sur Athena pour afficher le plan d'exécution de vos requêtes SQL. Utilisez `EXPLAIN ANALYZE` pour afficher le plan d'exécution distribué de vos requêtes SQL et le coût de chaque opération. Pour de plus amples informations, veuillez consulter Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena.
- Requêtes fédérées – Les requêtes fédérées sont prises en charge dans la version 2 du moteur Athena. Pour de plus amples informations, veuillez consulter Utilisation de la requête fédérée d'Amazon Athena.
- Fonctions géospatiales – Plus de 25 fonctions géospatiales ont été ajoutées. Pour de plus amples informations, veuillez consulter Nouvelles fonctions géospatiales de la version 2 du moteur Athena.
- Schéma imbriqué – La prise en charge de la lecture de schémas imbriqués a été ajoutée, réduisant ainsi les coûts.
- Instructions préparés – Utilisez des instructions préparées pour l'exécution répétée d'une même requête avec des paramètres de requête différents. Une déclaration préparée contient des paramètres fictifs dont vous transmettez les valeurs au moment de l'exécution. Les instructions préparées aident à prévenir les attaques par injection SQL. Pour de plus amples informations, veuillez consulter Utilisation de requêtes paramétrées.
- Prise en charge d'évolution de schéma – La prise en charge de l'évolution du schéma a été ajoutée pour les données au format Parquet.
  - Ajout de la prise en charge de la lecture de colonnes de type tableau, mappage ou ligne à partir de partitions dont le schéma de partition est différent du schéma de table. Cela peut se produire lorsque le schéma de table a été mis à jour après la création de la partition. Les types de colonnes modifiés doivent être compatibles. Pour les types ligne, des champs de fin peuvent être ajoutés ou supprimés, mais les champs correspondants (par ordre ordinal) doivent avoir le même nom.
  - Les fichiers ORC peuvent maintenant avoir des colonnes structurées avec des champs manquants. Cela permet de modifier le schéma de table sans réécrire les fichiers ORC.

  - Les colonnes structurées ORC sont maintenant mappées par le nom plutôt que par l'ordinal. Ceci permet de traiter correctement les champs structurés manquants ou supplémentaires dans le fichier ORC.
- SQL OFFSET – La clause SQL `OFFSET` est désormais prise en charge dans les instructions `SELECT`. Pour de plus amples informations, veuillez consulter SELECT.
- Instruction UNLOAD – Vous pouvez utiliser l'instruction `UNLOAD` pour écrire la sortie d'une requête `SELECT` aux formats PARQUET, ORC, AVRO et JSON. Pour de plus amples informations, veuillez consulter UNLOAD.

Amélioration des regroupements, des jointures et des sous-requêtes

- Regroupement complexe – Ajout de la prise en charge des opérations de regroupement complexes.
- Sous-requêtes corrélées – Ajout de la prise en charge des sous-requêtes corrélées dans les prédicats `IN` et des sous-requêtes corrélées nécessitant des forçages de type.
- CROSS JOIN – Ajout de la prise en charge de `CROSS JOIN` contre les tables dérivées `LATERAL`.
- JEUX DE REGROUPEMENT – Ajout de la prise en charge des clauses `ORDER BY` dans les agrégations pour les requêtes qui utilisent `GROUPING SETS`.
- Expressions Lambda – Ajout de la prise en charge du déréférencement des champs de ligne dans les expressions Lambda.
- Valeurs nulles dans les semi-jointures – Ajout de la prise en charge des valeurs nulles sur le côté gauche d'une semi-jointure (c'est-à-dire un prédicat `IN` avec des sous-requêtes).
- Jointures spatiales – Ajout de la prise en charge des jointures spatiales de diffusion et des jointures spatiales gauches.
- Déversement sur le disque – Pour les opérations `INNER JOIN` et `LEFT JOIN` à nécessitant beaucoup de mémoire, Athena décharge les résultats des opérations intermédiaires sur le disque. Cela permet l'exécution de requêtes nécessitant de grandes quantités de mémoire.

Amélioration des types de données

- INT pour INTEGER – Ajout de la prise en charge de `INT` comme alias pour le type de données `INTEGER`.
- Types INTERVAL – Ajout de la prise du forçage de type vers les types `INTERVAL`.

- IPADDRESS — Ajout d'un nouveau `IPADDRESS` type pour représenter les adresses IP dans les requêtes DML. Ajout de la prise en charge de la conversion entre le type `VARBINARY` et le type `IPADDRESS`. Le `IPADDRESS` type n'est pas reconnu dans les requêtes DDL.
- IS DISTINCT FROM – Ajout de la prise en charge de `IS DISTINCT FROM` pour le stypes `JSON` et `IPADDRESS`.
- Contrôles d'égalité des nuls – Les contrôles d'égalité pour les valeurs nulles dans les structures de données `ARRAY`, `MAP` et `ROW` sont désormais pris en charge. Par exemple, l'expression `ARRAY ['1', '3', null] = ARRAY ['1', '2', null]` renvoie `false`. Auparavant, un élément nul renvoyait le message d'erreur comparison not supported (comparaison non prise en charge).
- Forçage de type de rangée – La forçage entre les types de lignes sans tenir compte des noms de champs est maintenant autorisée. Auparavant, un type de ligne était forçable avec un autre type de ligne uniquement si le nom du champ dans le type source correspondait au type cible, ou si le type cible avait un nom de champ anonyme.
- Soustraction temporelle – Mise en œuvre de la soustraction pour tous les types `TIME` et `TIMESTAMP`.
- Unicode – Ajout de la prise en charge des séquences Unicode échappées dans les libellés chaînes.
- Concaténation VARBINY – Ajout de la prise en charge de la concaténation des valeurs `VARBINARY`.

  Fonctions de fenêtrage – Les fonctions de valeur de fenêtre prennent désormais en charge `IGNORE NULLS` et `RESPECT NULLS`.

Types d'entrée supplémentaires pour les fonctions

Les fonctions suivantes acceptent désormais des types d'entrée supplémentaires. Pour plus d'informations sur chaque fonction, consultez le lien correspondant à la documentation Presto.

- approx_distinct() – La fonction approx_distinct () prend désormais en charge les types suivants : `INTEGER`, `SMALLINT`, `TINYINT`, `DECIMAL`, `REAL`, `DATE`, `TIMESTAMP`, `TIMESTAMP WITH TIME ZONE`, `TIME`, `TIME WITH TIME ZONE`, `IPADDRESS` et `CHAR`.
- Avg(), sum() – Les fonctions d'agrégation avg() et sum() prennent désormais en charge le type de données `INTERVAL`.
- Lpad(), rpad() – Les fonctions lpad et rpad fonctionnent maintenant sur les entrées `VARBINARY`.

- Min(), max() – Les fonctions d'agrégation min() and max() autorisent désormais les types d'entrée inconnus au moment de l'analyse de la requête afin que vous puissiez utiliser les fonctions avec des libellés `NULL`.
- regexp_replace() – Ajout d'une variante de la fonction regexp_replace() qui peut exécuter une fonction Lambda pour chaque remplacement.
- Sequence() – Ajout de variantes `DATE` pour la fonction sequence(), y compris une variante avec un incrément implicite d'un jour.
- ST_Area() – La fonction géospatiale ST_Area() prend désormais en charge tous les types de géométrie.
- Substr() – La fonction substr fonctionne désormais sur les entrées `VARBINARY`.
- zip_with() – Les tableaux dont la longueur n'est pas adaptée peuvent maintenant être utilisés avec zip_with(). Les positions manquantes sont remplies avec le caractère nul. Auparavant, une erreur était générée lorsque des tableaux de longueurs différentes étaient transmis. Cette modification peut rendre difficile la distinction entre les valeurs qui étaient nulles à l'origine et celles qui ont été ajoutées pour compléter les tableaux à la même longueur.

Fonctions ajoutées

La liste suivante contient les fonctions qui sont nouvelles à partir de la version 2 du moteur Athena. La liste n'inclut pas les fonctions géospatiales. Pour une liste des fonctions géospatiales, voir Nouvelles fonctions géospatiales de la version 2 du moteur Athena.

Pour plus d'informations sur chaque fonction, consultez le lien correspondant à la documentation Presto.

Fonctions d’agrégation

reduce_agg ()

Opérateurs et fonctions de tableau

array_sort() – Ajout d'une variante de cette fonction qui prend une fonction Lambda comme comparateur.

ngrams ()

Fonctions et opérateurs binaires

from_big_endian_32()

from_ieee754_32()

from_ieee754_64()

hmac_md5()

hmac_sha1()

hmac_sha256()

hmac_sha512()

spooky_hash_v2_32()

spooky_hash_v2_64()

to_big_endian_32()q

to_ieee754_32()

to_ieee754_64()

Date and Time Functions and Operators

millisecond()

parse_duration()

to_milliseconds()

Mappage des fonctions et des opérateurs

multimap_from_entries()

Fonctions et opérateurs mathématiques

inverse_normal_cdf()

wilson_interval_lower()

wilson_interval_upper()

Fonctions de prétraitement quantile

Ajout des fonctions de prétraitement quantile et du type de prétraitement quantile `qdigest`.

Fonctions et opérateurs de chaîne

hamming_distance()

split_to_multimap()

Améliorations des performances

Les performances des fonctions suivantes ont été améliorées dans la version 2 du moteur Athena.

Les performances des requêtes

- Performance de jointure de diffusion – Amélioration des performances de jointure de diffusion en appliquant une taille dynamique de partition dans le composant master.
- Tables compartimentées – Amélioration des performances pour l'écriture dans des tables compartimentées lorsque les données écrites sont déjà partitionnées de manière appropriée (par exemple, lorsque la sortie provient d'une jointure compartimentée).
- DISTINCT – Amélioration des performances pour certaines requêtes utilisant `DISTINCT`.

  Filtrage dynamique et élagage des partitions – Ces améliorations permettent d'augmenter les performances et de réduire la quantité de données analysées dans les requêtes.
- Opérations de filtrage et projection – Les opérations de filtrage et de projection sont désormais toujours traitées par des colonnes si possible. Le moteur tire automatiquement parti des codages de dictionnaire lorsqu'ils sont efficaces.
- Regroupement des échanges – Amélioration des performances pour les requêtes avec regroupement des échanges.
- Agrégations globales – Amélioration des performances pour certaines requêtes qui effectuent des agrégations globales filtrées.
- JEUX DE REGROUPEMENT, CUBE, CUMUL – Amélioration des performances pour les requêtes impliquant `GROUPING SETS`, `CUBE` ou `ROLLUP`, que vous pouvez utiliser pour agréger plusieurs jeux de colonnes dans une seule requête.
- Filtres hautement sélectifs – Amélioration des performances des requêtes avec des filtres hautement sélectifs.
- Opérations JOIN et AGGREGATE – Amélioration des performances des opérations `JOIN` et `AGGREGATE`.
- LIKE – Amélioration des performances des requêtes qui utilisent des prédicats `LIKE` sur les colonnes des tables `information_schema`.

- ORDER BY (Ordonner par) et LIMIT (Limite) – Amélioration des plans, des performances et de l'utilisation de la mémoire pour les requêtes impliquant `ORDER BY` and `LIMIT` afin d'éviter les échanges de données inutiles.
- ORDER BY (Ordonner par) – Les opérations `ORDER BY` sont désormais distribuées par défaut, ce qui permet d'utiliser des clauses `ORDER BY` plus importantes.
- Conversions de type ROW – Amélioration des performances lors de la conversion entre les types `ROW`.
- Types structurels – Amélioration des performances des requêtes qui traitent des types structurels et qui contiennent des balayages, des jointures, des agrégations ou des écritures de table.
- Analyse de table – Une règle d'optimisation a été introduite pour éviter les analyses de tables en double dans certains cas.
- UNION – Amélioration des performances des requêtes `UNION`.

Performances de planification des requêtes

- Performances de planification – Amélioration des performances de planification des requêtes qui joignent plusieurs tables avec un grand nombre de colonnes.
- Évaluations des prédicats – Amélioration des performances d'évaluation des prédicats lors du refoulement des prédicats dans la planification.
- Prise en charge de refoulement de prédicat pour le forçage de type – Prise en charge du refoulement des prédicats pour le prédicat *<colonne>* `IN` *<liste de valeur>* où les valeurs de la liste de valeurs doivent être forcées pour correspondre au type de colonne.
- Inférence et refoulement des prédicats – Extension de l'inférence et du refoulement des prédicats et pour les requêtes qui utilisent un prédicat *<symbole>* `IN` *<sous-requête>*.
- Délais – Correction d'un bug qui pouvait, dans de rares cas, entraîner des délais d'expiration de la planification des requêtes.

Performances de jointure

- Jointures avec des colonnes mappées – Amélioration des performances des jointures et des agrégations qui incluent des colonnes mappées.
- Jointures avec uniquement des conditions de non-égalité – Amélioration des performances des jointures avec uniquement des conditions de non-égalité en utilisant une jointure par boucle imbriquée au lieu d'une jointure par hachage.

- Jointures externes – Le type de distribution de jointure est désormais automatiquement sélectionné pour les requêtes impliquant des jointures externes.
- Jointure de plage sur une fonction – Amélioration des performances des jointures lorsque la condition est une plage sur une fonction (par exemple, `a JOIN b ON b.x < f(a.x) AND b.x > g(a.x)`).
- S pill-to-disk — Correction de bogues et de problèmes de mémoire spill-to-disk connexes afin d'améliorer les performances et de réduire les erreurs de mémoire lors `JOIN` des opérations.

Performances des sous-requêtes

- Sous-requêtes EXISTS corrélées – Amélioration des performances des sous-requêtes `EXISTS` corrélées.
- Sous-requêtes corrélées avec des égalités – Amélioration de la prise en charge des sous-requêtes corrélées contenant des prédicats d'égalité.
- Sous-requêtes corrélées avec des inégalités – Amélioration des performances des sous-requêtes corrélées qui contiennent des inégalités.
- Agrégations Count(*) sur les sous-requêtes – Amélioration des performances des agrégations `count(*)` sur des sous-requêtes avec une cardinalité constante connue.
- Propagation du filtre de requête externe – Amélioration des performances des sous-requêtes corrélées lorsque les filtres de la requête externe peuvent être propagés à la sous-requête.

Performance des fonctions

- Fonctions de fenêtrage d'agrégation – Amélioration des performances des fonctions de fenêtrage d'agrégation.
- element_at() – Amélioration des performances de `element_at()` pour les mappages afin que le temps soit constant plutôt que proportionnel à la taille du mappage.
- Grouping() – Amélioration des performances des requêtes impliquant `grouping()`.
- Forçage JSON – Amélioration des performances du forçage des types `JSON` à `ARRAY` or `MAP`.
- Fonctions de retour de mappage – Amélioration des performances des fonctions de retour de mappage.
- ap-to-map Casting M — Amélioration des performances du map-to-map casting.
- Min() et max() – Les fonctions `min()` et `max()` ont été optimisées pour éviter la création inutile d'objets, réduisant ainsi les coûts du récupérateur de mémoire.

- row_number() – Amélioration des performances et de l'utilisation de la mémoire pour les requêtes utilisant `row_number()` suivi d'un filtre sur les numéros de ligne générés.
- Fonctions de fenêtrage – Amélioration des performances des requêtes contenant des fonctions de fenêtrage avec des clauses `PARTITION BY` et `ORDER BY` identiques.
- Fonctions de fenêtrage – Amélioration des performances de certaines fonctions de fenêtrage (par exemple, `LAG`) qui ont des spécifications similaires.

Performances géospatiales

- Sérialisation de la géométrie – Amélioration des performances de sérialisation des valeurs géométriques.
- Fonctions géospatiales – Amélioration des performances de `ST_Intersects()`, `ST_Contains()`, `ST_Touches()`, `ST_Within()`, `ST_Overlaps()`, `ST_Disjoint()`, `transform_values()`, `ST_XMin()`, `ST_XMax()`, `ST_YMin()`, `ST_YMax()`, `ST_Crosses()` et `array_intersect()`.
- ST_Distance() – Amélioration des performances des requêtes de jointure impliquant la fonction `ST_Distance()`.
- ST_Intersection() – Optimisation de la fonction `ST_Intersection()` pour les rectangles alignés sur les axes de coordonnées (par exemple, les polygones produits par les fonctions `ST_Envelope()` et `bing_tile_polygon()`).

Améliorations relatives à JSON

Fonctions de mappage

- Amélioration des performances de l'indice de mappage de `O(n)` à `O(1)` dans tous les cas. Auparavant, seules les mappages produits par certaines fonctions et lecteurs ont profité de cette amélioration.
- Ajout des fonctions `map_from_entries()` et `map_entries()`.

Forçage de type

- Ajout de la possibilité de forçer vers `JSON` à partir de `REAL`, `TINYINT` ou `SMALLINT`.
- Vous pouvez désormais forçer `JSON` vers `ROW` même si le code `JSON` ne contient pas tous les champs de la `ROW`.

- Amélioration des performances de `CAST(json_parse(...) AS ...)`.
- Amélioration des performances du forçage du type `JSON` au type `ARRAY` ou `MAP`.

Nouvelles fonctions JSON

- is_json_scalar()

Évolutions

Les modifications de type interruption comprennent des corrections de bogues, des modifications des fonctions géospatiales, des fonctions remplacées et l'introduction de limites. Les améliorations de la conformité à ANSI SQL peuvent interrompre les requêtes qui dépendaient d'un comportement non standard.

Correctifs de bogue

Les modifications suivantes corrigent des problèmes de comportement qui entraînaient l'exécution réussie de requêtes, mais avec des résultats imprécis.

- Les colonnes parquet fixed_len_byte_array sont désormais acceptées en DECIMAL – Les requêtes sur des colonnes Parquet de type `fixed_len_byte_array` réussissent et renvoient des valeurs correctes si elles sont annotées comme `DECIMAL` dans le schéma Parquet. Les requêtes sur des colonnes `fixed_len_byte_array` sans annotation `DECIMAL` échouent avec une erreur. Auparavant, les requêtes sur des colonnes `fixed_len_byte_array` sans annotation DECIMAL réussissaient, mais renvoyaient des valeurs incompréhensibles.
- json_parse() n'ignore plus les caractère de droite – Auparavant, les entrées telles que `[1,2]abc` étaient analysées avec succès en tant que `[1,2]`. L'utilisation de caractères de droite produit maintenant le message d'erreur Cannot convert '[1, 2]abc' to JSON (Impossible de convertir '[1, 2]abc' en JSON).
- Précision décimale Round() corrigée – `round(x, d)` arrondit maintenant correctement `x` quand `x` est un DECIMAL ou quand `x` est un DECIMAL avec l'échelle 0 et `d` est un entier négatif. Auparavant, aucun arrondissement n'était effectué dans ces cas.
- round(x, d) et truncate(x, d) – Le paramètre `d` dans la signature des fonctions `round(x, d)` et `truncate(x, d)` est maintenant de type `INTEGER`. Précédemment, `d` pourrait être de type `BIGINT`.
- Map() avec des clés dupliquées – `map()` déclenche désormais une erreur en cas de clés dupliquées au lieu de produire silencieusement un mappage corrompu. Les requêtes qui

construisent actuellement des valeurs de mappage en utilisant des clés dupliquées échouent désormais avec une erreur.

- map_from_entries() déclenche une erreur avec des entrées nulles – `map_from_entries()` déclenche désormais une erreur lorsque le tableau d'entrée contient une entrée nulle. Requêtes qui construisent un mappage en passant `NULL` comme une valeur échouent maintenant.
- Tables – Les tables qui ont des types de partition non pris en charge ne peuvent plus être créées.
- Amélioration de la stabilité numérique des fonctions statistiques – La stabilité numérique des fonctions statistiques `corr()`, `covar_samp()`, `regr_intercept()` et `regr_slope()` a été améliorée.
- La précision TIMESTAMP (Horodatage) définie dans parquet est désormais respectée – La précision des valeurs `TIMESTAMP` et la précision définie pour la colonne `TIMESTAMP` dans le schéma Parquet doivent désormais correspondre. Les précisions non correspondantes entraînent des horodatages incorrects.
- Informations de fuseau horaire – Les informations relatives aux fuseaux horaires sont désormais calculées à l'aide du progiciel java.time du kit SDK Java 1.8.
- SUM (Somme) des types de données INTERCAL_DAY_TO_SECOND et INTERCAL_YEAR_TO_MONTH – Vous ne pouvez plus utiliser `SUM(NULL)` directement. Pour utiliser `SUM(NULL)`, forcez `NULL` vers un type de données comme `BIGINT`, `DECIMAL`, `REAL`, `DOUBLE`, `INTERVAL_DAY_TO_SECOND` ou `INTERVAL_YEAR_TO_MONTH`.

Modifications apportées aux fonctions géospatiales

Les modifications apportées aux fonctions géospatiales sont les suivantes.

- Modifications des noms des fonctions – Certains noms de fonctions ont changé. Pour de plus amples informations, veuillez consulter Modifications des noms des fonctions géospatiales dans la version 2 du moteur Athena.
- VARBINARY – Le type `VARBINARY` n'est plus directement pris en charge pour l'entrée des fonctions géospatiales. Par exemple, pour calculer directement la surface d'une géométrie, celle-ci doit maintenant être saisie au format `VARCHAR` ou `GEOMETRY`. La solution de contournement consiste à utiliser les fonctions de transformation, comme dans les exemples suivants.
  - Pour utiliser `ST_area()` afin de calculer la zone pour une entrée `VARBINARY` au format WKB (Well-Known Binary), passez d'abord l'entrée à `ST_GeomFromBinary()`, par exemple :

```
ST_area(ST_GeomFromBinary(<wkb_varbinary_value>))
```

- Pour utiliser `ST_area()` afin de calculer la zone pour une entrée `VARBINARY` au format binaire hérité, passez d'abord l'entrée à la fonction `ST_GeomFromLegacyBinary()`, par exemple :

```
ST_area(ST_GeomFromLegacyBinary(<legacy_varbinary_value>))
```

- ST_ ExteriorRing () et ST_Polygon () — `ST_ExteriorRing()`et n'acceptent `ST_Polygon()`désormais que les polygones en entrée. Auparavant, ces fonctions acceptaient par erreur d'autres géométries.
- ST_Distance() – Comme l'exige la spécification SQL/MM, la fonction `ST_Distance()` renvoie maintenant `NULL` si l'une des entrées est une géométrie vide. Auparavant, `NaN` était renvoyé.

Conformité ANSI SQL

Les problèmes de syntaxe et de comportement suivants ont été corrigés pour respecter la norme ANSI SQL.

- Opérations cast() – Les opérations cast() de REAL ou DOUBLE vers DECIMAL sont désormais conformes à la norme SQL. Par exemple, `cast (double '10000000000000000000000000000000' as decimal(38))` retournait auparavant `10000000000000000536616220439347 2`, mais retourne maintenant `10000000000000000000000000000000`.
- JOIN ... USING – `JOIN ... USING` est maintenant conforme à la sémantique SQL standard. Auparavant, `JOIN ... USING` nécessitait de qualifier le nom de la table en colonnes, et la colonne des deux tables était présente dans le résultat. Les qualifications de la table sont maintenant invalides et la colonne n'est présente qu'une seule fois dans le résultat.
- Suppression des libellés de type ROW – Le format de libellé `ROW<int, int>(1, 2)` du type ROW n'est plus pris en charge. Utilisez la syntaxe `ROW(1 int, 2 int)` à la place.
- Sémantique d'agrégation groupée – Les agrégations groupées utilisent la sémantique `IS NOT DISTINCT FROM` plutôt que sémantique d'égalité. LLes agrégations groupées renvoient désormais des résultats corrects et présentent des performances améliorées lors du regroupement sur des valeurs à virgule flottante `NaN`. Le regroupement sur les types de mappage, de liste et de ligne qui contiennent des valeurs nulles est pris en charge.
- Les types avec guillemets ne sont plus autorisés – Conformément à la norme ANSI SQL, les types de données ne peuvent plus être placés entre guillemets. Par exemple, `SELECT "date" '2020-02-02'` n'est plus une requête valide. À la place, utilisez la syntaxe `SELECT date '2020-02-02'`.

- Accès aux champs de ligne anonymes – Les champs de ligne anonymes ne sont plus accessibles à l'aide de la syntaxe `[.field0, .field1, ...]`.
- Opérations de regroupement complexes – Les opérations de regroupement complexes, `GROUPING SETS`, `CUBE` et `ROLLUP` ne prennent pas en charge le regroupement sur des expressions composées de colonnes d'entrée. Seuls les noms de colonnes sont autorisés.

Fonctions remplacées

Les fonctions suivantes ne sont plus prises en charge et ont été remplacées par une syntaxe qui produit les mêmes résultats.

- information_schema.__internal_partitions__ – L'utilisation de `__internal_partitions__` n'est plus prise en charge par le moteur Athena version 2. Pour une syntaxe équivalente, utilisez `SELECT * FROM "<table_name>$partitions"` ou `SHOW PARTITIONS`. Pour de plus amples informations, veuillez consulter Liste des partitions d'une table spécifique.
- Fonctions géospatiales remplacées – Pour une liste des fonctions géospatiales dont les noms ont changé, voirModifications des noms des fonctions géospatiales dans la version 2 du moteur Athena.

Limites

Les limites suivantes ont été introduites dans la version 2 du moteur Athena afin de s'assurer que les requêtes n'échouent pas en raison de limitations de ressources. Ces limites ne sont pas configurables par les utilisateurs.

- Nombre d'éléments de résultat – Le nombre d'éléments de résultat n est limité à 10 000 ou moins pour les fonctions suivantes : `min(col, n)`, `max(col, n)`, `min_by(col1, col2, n)` et `max_by(col1, col2, n)`.
- JEUX DE GROUPES – Le nombre maximal de tranches dans un jeu de groupes est 2048.
- Longueur maximale de la ligne du fichier texte : la longueur de ligne maximale par défaut pour les fichiers texte est de 200 Mo.
- Taille maximale des résultats de la fonction de séquence – La taille maximale de résultat d'une fonction de séquence est de 50 000 entrées. Par exemple, `SELECT sequence(0,45000,1)` réussit, mais `SELECT sequence(0,55000,1)` échoue avec le message d'erreur Le résultat de la fonction de séquence ne doit pas comporter plus de 50 000 entrées. Cette limite s'applique à tous les types d'entrées pour les fonctions de séquence, y compris aux horodatages.

Référence SQL pour Athena

Amazon Athena prend en charge un sous-ensemble d'instructions, de fonctions, d'opérateurs et de types de données en langage de définition de données (DDL) et en langage de manipulation de données (DML). À quelques exceptions près, Athena DDL est basé sur HiveQL DDL et Athena DML est basé sur Trino. Pour plus d'informations sur les versions du moteur Athena, voir Gestion des versions du moteur Athena.

Rubriques

Types de données dans Amazon Athena

Lorsque vous exécutez`CREATE TABLE`, vous spécifiez les noms des colonnes et le type de données que chaque colonne peut contenir. Les tables que vous créez sont stockées dans le AWS Glue Data Catalog.

Pour faciliter l'interopérabilité avec d'autres moteurs de requêtes, Athena utilise les noms de types de données Apache Hive pour les instructions DDL telles que. `CREATE TABLE` Pour les requêtes DML telles que`SELECT`,, et `CTASINSERT INTO`, Athena utilise les noms des types de données Trino. Le tableau suivant indique les types de données pris en charge dans Athena. Lorsque les types DDL et DML diffèrent en termes de nom, de disponibilité ou de syntaxe, ils sont indiqués dans des colonnes distinctes.

DDL	DML	Description
BOOLEAN		Les valeurs sont `true` et `false`.
TINYINT		Un entier signé de 8 bits au format de complément à deux, avec une valeur minimale de -2 ^{7} et une valeur maximale de 2 ^{7} -1.

DDL	DML	Description
SMALLINT		Un entier signé de 16 bits au format de complément à deux, avec une valeur minimale de -2 ^{15} et une valeur maximale de 2 ^{15} -1.
INT, ENTIER		Une valeur signée de 32 bits au format de complément à deux, avec une valeur minimale de -2 ^{31} et une valeur maximale de 2 ^{31} -1.
BIGINT		Un entier signé de 64 bits au format de complément à deux, avec une valeur minimale de -2 ^{63} et une valeur maximale de 2 ^{63} -1.
FLOAT	REAL	Nombre à virgule flottante à précision unique signé sur 32 bits. La plage est comprise entre 1,40129846432481707e-45 et 3,40282346638528860e+38, positif ou négatif. Conforme à la norme IEEE pour l'arithmétique à virgule flottante (IEEE 754).
DOUBLE		Nombre à virgule flottante à double précision signé sur 64 bits. La plage va de 4,94065645841246544e-324d à 1,79769313486231570e+308d, positif ou négatif. Conforme à la norme IEEE pour l'arithmétique à virgule flottante (IEEE 754).
DECIMAL (*précision* , *échelle*)		*precision* est le nombre total de chiffres. *scale*(facultatif) est le nombre de chiffres dans la partie fractionnaire avec une valeur par défaut de 0. Par exemple, utilisez ces définitions de type : `decimal(11,5)` , `decimal(15)` . La valeur maximale pour la *précision* est de 38, et la valeur maximale pour l'*échelle* est de 38.

DDL	DML	Description
CHAR, CHAR (*longueur*)		Données de caractères de longueur fixe, avec une longueur spécifiée comprise entre 1 et 255, telles que char (10). Si *la longueur* est spécifiée, les chaînes sont tronquées à la longueur spécifiée lors de la lecture. Si la chaîne de données sous-jacente est plus longue, elle reste inchangée. Pour plus d'informations, consultez la section relative au type de données Hive CHAR.
CHAÎNE	VARCHAR	Données de caractères de longueur variable.
VARCHAR (longueur)		Données de caractères de longueur variable avec une longueur de lecture maximale. Les chaînes sont tronquées à la longueur spécifiée lors de la lecture. Si la chaîne de données sous-jacente est plus longue, elle reste inchangée.
BINAIRE	VARBINARY	Données binaires de longueur variable.
TIME		Un moment de la journée précis à la milliseconde.
Non disponible	HEURE (*précision*)	Un moment de la journée avec une précision précise. `TIME(3)`est équivalent à`TIME`.
Non disponible	TIME WITH TIME ZONE	Heure de la journée dans un fuseau horaire. Les fuseaux horaires doivent être spécifiés sous forme de décalages par rapport à l'UTC.
DATE		Une date calendaire avec l'année, le mois et le jour.
TIMESTAMP	HORODATAGE, HORODATAGE SANS FUSEAU HORAIRE	Date et heure calendaires précises à la milliseconde.

DDL	DML	Description
Non disponible	HORODATAGE (*précision*), HORODATAGE (*précision*) SANS FUSEAU HORAIRE	Une date et une heure calendaires avec une précision précise. `TIMESTAMP(3)` est équivalent à`TIMESTAMP` .
Non disponible	TIMESTAMP WITH TIME ZONE	Date et heure calendaires dans un fuseau horaire. Les fuseaux horaires peuvent être spécifiés sous forme de décalages par rapport à l'UTC, sous forme de noms de fuseaux horaires IANA ou en utilisant UTC, UT, Z ou GMT.
Non disponible	HORODATAGE (*précision*) AVEC FUSEAU HORAIRE	Une date et une heure calendaires avec une précision précise, dans un fuseau horaire.
Non disponible	INTERVALLE D'UNE ANNÉE À L'AUTRE	Un intervalle d'un ou plusieurs mois entiers
Non disponible	INTERVALLE D'UN JOUR À L'AUTRE	Intervalle d'une ou plusieurs secondes, minutes, heures ou jours
TABLEAU < type_élément >	TABLEAU [*type_élément*]	Un tableau de valeurs. Toutes les valeurs doivent être du même type.
MAP< type_clé, type_valeur >	*CARTE (type_clé, type_valeur)*	Une carte où les valeurs peuvent être recherchées par clé. Toutes les clés doivent avoir la même valeur, et toutes les valeurs doivent avoir la même valeur.

DDL	DML	Description
STRUCT< nom_champ_1 : type_champ_1, nom_champ_2 : type_champ_2 ,... >	*LIGNE (nom_champ_1 type_champ_1, nom_champ_2 type_champ_2 ,...)*	Structure de données avec des champs nommés et leurs valeurs.
Non disponible	JSON	Type de valeur JSON, qui peut être un objet JSON, un tableau JSON, un numéro JSON, une chaîne JSON `false` ou`null`. `true`
Non disponible	UUID	Un UUID (Universal Unique IDentifier).
Non disponible	ADRESSE IP	Une adresse IPv4 ou IPv6.
Non disponible	HyperLogLog	Ces types de données prennent en charge des fonctions internes approximatives. Pour plus d'informations sur chaque type, consultez le lien vers l'entrée correspon dante dans la documentation de Trino.
	P4 HyperLogLog	
	SetDigest	
	QDigest	
	TDigest	

Exemples de types de données

Le tableau suivant présente des exemples de littéraux pour les types de données DML.

Type de données	Exemples
BOOLEAN	`true` `false`

Type de données	Exemples
TINYINT	`TINYINT '123'`
SMALLINT	`SMALLINT '123'`
INT, ENTIER	`123456790`

BIGINT	`BIGINT '1234567890'`
	
	`2147483648`

REAL	`'123456.78'`
DOUBLE	`1.234`
DECIMAL (*précision* , *échelle*)	`DECIMAL '123.456'`
CHAR, CHAR (*longueur*)	`CHAR 'hello world'`, `CHAR 'hello ''world''!'`
VARCHAR, VARCHAR (longueur)	`VARCHAR 'hello world'`, `VARCHAR 'hello ''world''!'`
VARBINARY	`X'00 01 02'`
HEURE, HEURE (*précision*)	`TIME '10:11:12'` , `TIME '10:11:12.345'`
TIME WITH TIME ZONE	`TIME '10:11:12.345 -06:00'`
DATE	`DATE '2024-03-25'`
HORODATAGE, HORODATAGE SANS FUSEAU HORAIRE, HORODATAGE (précision), HORODATAGE (précision) SANS FUSEAU HORAIRE	`TIMESTAMP '2024-03-25 11:12:13'` , `TIMESTAMP '2024-03-25 11:12:13.456'`

Type de données	Exemples
HORODATAGE AVEC FUSEAU HORAIRE, HORODATAGE (*précision*) AVEC FUSEAU HORAIRE	`TIMESTAMP '2024-03-25 11:12:13.456 Europe/Berlin'`
INTERVALLE D'UNE ANNÉE À L'AUTRE	`INTERVAL '3' MONTH`
INTERVALLE D'UN JOUR À L'AUTRE	`INTERVAL '2' DAY`
TABLEAU [*type_élément*]	`ARRAY['one', 'two', 'three']`
CARTE (type_clé, type_valeur)	`MAP(ARRAY['one', 'two', 'three'], ARRAY[1, 2, 3])` Notez que les cartes sont créées à partir d'un tableau de clés et d'un tableau de valeurs.

LIGNE (nom_champ_1 type_champ_1, nom_champ_2 type_champ_2 ,...)	`ROW('one', 'two', 'three')`
	
	Notez que les lignes créées de cette façon n'ont aucun nom de colonne. Pour ajouter des noms de colonnes, vous pouvez utiliserCAST, comme dans l'exemple suivant :
	
	`CAST(ROW(1, 2, 3) AS ROW(one INT, two INT, three INT))`

JSON	`JSON '{"one":1, "two": 2, "three": 3}'`
UUID	`UUID '12345678-90ab-cdef-1234-567890abcdef'`

ADRESSE IP	`IPADDRESS '10.0.0.1'`
	
	`IPADDRESS '2001:db8::1'`

Considérations relatives aux types de données

Limites de taille

Pour les types de données qui ne spécifient pas de limite de taille, gardez à l'esprit qu'il existe une limite pratique de 32 Mo pour toutes les données d'une seule ligne. Pour plus d'informations, consultez Row or column size limitation dans Considérations et limitations relatives aux requêtes SQL dans Amazon Athena.

CHAR et VARCHAR

Une `CHAR(n)` valeur comporte toujours un nombre de *n* caractères. Par exemple, si vous convertissez « abc » en`CHAR(7)`, 4 espaces de fin sont ajoutés.

Les comparaisons de `CHAR` valeurs incluent les espaces de début et de fin.

Si une longueur est spécifiée pour `CHAR` ou`VARCHAR`, les chaînes sont tronquées à la longueur spécifiée lors de la lecture. Si la chaîne de données sous-jacente est plus longue, elle reste inchangée.

Pour éviter un guillemet unique dans un `CHAR` ou`VARCHAR`, utilisez un guillemet unique supplémentaire.

Pour convertir un type de données autre qu'une chaîne en chaîne dans une requête DML, convertissez-le en type de `VARCHAR` données.

Pour utiliser la `substr` fonction afin de renvoyer une sous-chaîne d'une longueur spécifiée à partir d'un type de `CHAR` données, vous devez d'abord convertir la `CHAR` valeur en`VARCHAR`. Dans l'exemple suivant, `col1` utilise le type de `CHAR` données.

```
substr(CAST(col1 AS VARCHAR), 1, 4)
```

DECIMAL

Pour spécifier des valeurs décimales sous forme de littéraux dans les `SELECT` requêtes, par exemple lorsque vous sélectionnez des lignes contenant une valeur décimale spécifique, vous pouvez spécifier le `DECIMAL` type et répertorier la valeur décimale sous forme de littéral entre guillemets simples dans votre requête, comme dans les exemples suivants.

```
SELECT * FROM my_table
```

```
WHERE decimal_value = DECIMAL '0.12'
```

```
SELECT DECIMAL '44.6' + DECIMAL '77.2'
```

Utilisation des données d'horodatage

Cette section décrit certaines considérations relatives à l'utilisation des données d'horodatage dans Athena.

> **Note**
>
> Le traitement des horodatages a quelque peu changé entre les versions 2 et 3 du moteur Athena. Pour plus d'informations sur les erreurs liées à l'horodatage qui peuvent se produire dans la version 3 du moteur Athena et sur les solutions proposées, consultez Modifications d'horodatage dans la référence Version 3 du moteur Athena.

Format pour écrire des données d'horodatage dans des objets Amazon S3

Le format dans lequel les données d'horodatage doivent être écrites dans les objets Amazon S3 dépend à la fois du type de données de colonne et de la SerDebibliothèque que vous utilisez.

- Si vous avez une colonne de table de type `DATE`, Athena s'attend à ce que la colonne ou la propriété correspondante des données soit une chaîne au format ISO `YYYY-MM-DD` ou un type de date intégré comme ceux de Parquet ou ORC.
- Si vous avez une colonne de table de type `TIME`, Athena s'attend à ce que la colonne ou la propriété correspondante des données soit une chaîne au format ISO `HH:MM:SS` ou un type d'heure intégré comme ceux de Parquet ou ORC.
- Si vous avez une colonne de table de type `TIMESTAMP`, Athena s'attend à ce que la colonne ou la propriété correspondante des données soit une chaîne au format `YYYY-MM-DD HH:MM:SS.SSS` (notez l'espace entre la date et l'heure) ou un type d'heure intégré comme ceux de Parquet, ORC ou Ion.

> **Note**
>
> Les horodatages SerDe OpenCSV constituent une exception et doivent être codés sous forme d'époques UNIX d'une résolution de l'ordre de la milliseconde.

S'assurer que les données partitionnées dans le temps correspondent au champ d'horodatage d'un enregistrement

Le producteur des données doit s'assurer que les valeurs de partition correspondent aux données contenues dans la partition. Par exemple, si vos données ont une `timestamp` propriété et que vous utilisez Firehose pour les charger dans Amazon S3, vous devez utiliser le partitionnement dynamique car le partitionnement par défaut de Firehose est le suivant. wall-clock-based

Utiliser STRING comme type de données pour les clés de partition

Pour des raisons de performances, il est préférable d'utiliser `STRING` comme type de données pour les clés de partition. Même si Athena reconnaît les valeurs de partition au format `YYYY-MM-DD` comme des dates lorsque vous utilisez le type `DATE`, cela peut entraîner de mauvaises performances. Pour cette raison, nous vous recommandons d'utiliser plutôt le type de données `STRING` pour les clés de partition.

Comment écrire des requêtes pour des champs d'horodatage qui sont également partitionnés dans le temps

La façon dont vous rédigez les requêtes pour les champs d'horodatage partitionnés dans le temps dépend du type de table que vous souhaitez interroger.

Tables Hive

Avec les tables Hive les plus couramment utilisées dans Athena, le moteur de requête n'a aucune connaissance des relations entre les colonnes et les clés de partition. Pour cette raison, vous devez toujours ajouter des prédicats dans vos requêtes pour la colonne et pour la clé de partition.

Supposons, par exemple, que vous disposiez d'une colonne `event_time` et d'une clé de partition `event_date` et que vous souhaitiez interroger des événements survenus entre 23 h 00 et 03 h 00. Dans ce cas, vous devez inclure des prédicats dans votre requête pour la colonne et pour la clé de partition, comme dans l'exemple suivant.

```
WHERE event_time BETWEEN start_time AND end_time
  AND event_date BETWEEN start_time_date AND end_time_date
```

Tables Iceberg

Avec les tables Iceberg, vous pouvez utiliser des valeurs de partition calculées, ce qui simplifie vos requêtes. Supposons, par exemple, que votre table Iceberg ait été créée avec une clause `PARTITIONED BY` comme celle-ci :

```
PARTITIONED BY (event_date month(event_time))
```

Dans ce cas, le moteur de requête réduit automatiquement les partitions en fonction des valeurs des prédicats `event_time`. Pour cette raison, votre requête doit uniquement spécifier un prédicat pour `event_time`, comme dans l'exemple suivant.

```
WHERE event_time BETWEEN start_time AND end_time
```

Pour plus d'informations, voir Création de tables Iceberg.

Requêtes, fonctions et opérateurs DML

Le moteur de requêtes Athena DML prend généralement en charge les syntaxes Trino et Presto et apporte ses propres améliorations. Athena ne prend pas en charge toutes les fonctionnalités Trino ou Presto. Pour plus d'informations, consultez les rubriques relatives aux instructions spécifiques de cette section et Considérations et restrictions. Pour plus d'informations sur les fonctions, veuillez consulter la rubrique Fonctions dans Amazon Athena. Pour plus d'informations sur les versions du moteur Athena, voir Gestion des versions du moteur Athena.

Pour plus d'informations sur les instructions DDL, voir Instructions DDL. Pour une liste des instructions DDL non prises en charge, voir DDL non prise en charge.

SELECT

Récupère les lignes de données de zéro ou plusieurs tables.

> **Note**
>
> Cette rubrique fournit des informations récapitulatives à titre de référence. Cette documentation n'a pas pour objectif de couvrir en détail l'utilisation de `SELECT` et du langage SQL. Pour des informations sur l'utilisation de SQL spécifique à Athena, voir Considérations et limitations relatives aux requêtes SQL dans Amazon Athena et Exécution de requêtes SQL à l'aide d'Amazon Athena. En guise d'exemple en matière de création d'une base de données, de création d'une table et d'exécution d'une requête `SELECT` sur la table dans Athena, voir Mise en route.

Résumé

```
[ WITH with_query [, ...] ]
SELECT [ ALL | DISTINCT ] select_expression [, ...]
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
[ HAVING condition ]
[ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select ]
[ ORDER BY expression [ ASC | DESC ] [ NULLS FIRST | NULLS LAST] [, ...] ]
[ OFFSET count [ ROW | ROWS ] ]
[ LIMIT [ count | ALL ] ]
```

> **Note**
>
> Les mots réservés dans les instructions SQL SELECT doivent être placés entre guillemets doubles. Pour plus d'informations, consultez Liste des mots-clés réservés dans des instructions SQL SELECT.

Paramètres

[WITH with_query [, ...]]

Vous pouvez utiliser `WITH` pour aplatir les requêtes imbriquées ou pour simplifier les sous-requêtes.

L'utilisation de la clause `WITH` pour créer des requêtes récursives est prise en charge à partir de la version 3 du moteur Athena. La profondeur de récursivité maximale est de 10.

La clause `WITH` précède la liste `SELECT` dans une requête et définit une ou plusieurs sous-requêtes pour une utilisation au sein de la requête `SELECT`.

Chaque sous-requête définit une table temporaire, similaire à la définition d'une vue, que vous pouvez référencer dans la clause `FROM`. Les tables sont utilisées uniquement lorsque la requête s'exécute.

La syntaxe de `with_query` est la suivante :

```
subquery_table_name [ ( column_name [, ...] ) ] AS (subquery)
```

Où :

- `subquery_table_name` est un nom unique d'une table temporaire qui définit les résultats de la sous-requête de la clause `WITH`. Chaque `subquery` doit avoir un nom de table qui peut être référencé dans la clause `FROM`.
- `column_name [, ...]` est une liste facultative de noms de colonne de sortie. Le nombre de noms de colonne doit être égal ou inférieur au nombre de colonnes défini par `subquery`.
- `subquery` désigne n'importe quelle instruction de requête.

[TOUS | DISTINCT] select_expression

`select_expression` détermine les lignes à sélectionner. A `select_expression` peut utiliser l'un des formats suivants :

```
expression [ [ AS ] column_alias ] [, ...]
```

```
row_expression.* [ AS ( column_alias [, ...] ) ]
```

```
relation.*
```

```
*
```

- La `expression [ [ AS ] column_alias ]` syntaxe spécifie une colonne de sortie. La `[AS] column_alias` syntaxe facultative spécifie un nom de titre personnalisé à utiliser pour la colonne dans la sortie.
- Pour`row_expression.* [ AS ( column_alias [, ...] ) ]`, `row_expression` est une expression arbitraire du type de données`ROW`. Les champs de la ligne définissent les colonnes de sortie à inclure dans le résultat.
- En effet`relation.*`, les colonnes de `relation` sont incluses dans le résultat. Cette syntaxe n'autorise pas l'utilisation d'alias de colonne.
- L'astérisque `*` indique que toutes les colonnes doivent être incluses dans le jeu de résultats.
- Dans le jeu de résultats, l'ordre des colonnes est identique à l'ordre de leur spécification par l'expression de sélection. Si une expression de sélection renvoie plusieurs colonnes, l'ordre des colonnes suit l'ordre utilisé dans la relation source ou l'expression de type ligne.
- Lorsque des alias de colonne sont spécifiés, ils remplacent les noms de champs de colonne ou de ligne préexistants. Si l'expression select ne comporte pas de nom de colonne, les noms de colonnes anonymes indexés à zéro (`_col0`,`_col1`,`_col2, ...`) sont affichés dans la sortie.

- ALL est la valeur par défaut. L'utilisation d'ALL est traité de la même façon que si la valeur avait été omise ; toutes les lignes de toutes les colonnes sont sélectionnées et les doublons sont conservés.
- Utilisez DISTINCT pour renvoyer uniquement des valeurs distinctes lorsqu'une colonne contient des valeurs en double.

FROM from_item [, ...]

Indique les entrées de la requête, où from_item peut être une vue, une construction de jointure ou une sous-requête comme décrit ci-dessous.

L'élément from_item peut être l'un ou l'autre :

- table_name [[AS] alias [(column_alias [, ...])]]

  Où table_name est le nom de la table cible à partir de laquelle sélectionner les lignes, où alias est le nom pour donner la sortie de l'instruction SELECT et où column_alias définit les colonnes de l'alias spécifié.

-OU-

- join_type from_item [ON join_condition | USING (join_column [, ...])]

  Où join_type est l'un des éléments suivants :

  - [INNER] JOIN
  - LEFT [OUTER] JOIN
  - RIGHT [OUTER] JOIN
  - FULL [OUTER] JOIN
  - CROSS JOIN
  - ON join_condition | USING (join_column [, ...]) Où l'utilisation de join_condition vous permet de spécifier les noms de colonne pour les clés de jointure de plusieurs tables et où l'utilisation de join_column nécessite join_column pour exister dans les deux tables.

[Condition WHERE]

Filtre les résultats en fonction de la condition que vous spécifiez, où condition a généralement la syntaxe suivante.

```
column_name operator value [[[AND | OR] column_name operator value] ...]
```

L'*opérateur* peut être l'un des comparateurs =, >, <, >=, <=, <>, !=.

Les expressions de sous-requêtes suivantes peuvent également être utilisées dans la clause WHERE.

- [NOT] BETWEEN *integer_A* AND *integer_B* – Spécifie une plage entre deux entiers, comme dans l'exemple suivant. Si le type de données de colonne est varchar, la colonne doit d'abord être convertie en entier.

```
SELECT DISTINCT processid FROM "webdata"."impressions"
WHERE cast(processid as int) BETWEEN 1500 and 1800
ORDER BY processid
```

- [NOT] LIKE *value* – Recherche le motif spécifié. Utilisez le signe de pourcentage (%) comme caractère générique, comme dans l'exemple suivant.

```
SELECT * FROM "webdata"."impressions"
WHERE referrer LIKE '%.org'
```

- [NOT] IN (*value*[, *value*[, ...]]) – Spécifie une liste de valeurs possibles pour une colonne, comme dans l'exemple suivant.

```
SELECT * FROM "webdata"."impressions"
WHERE referrer IN ('example.com','example.net','example.org')
```

[GROUP BY [ALL | DISTINCT] grouping_expressions [, ...]]

Divise la sortie de l'instruction SELECT en lignes avec les valeurs correspondantes.

ALL et DISTINCT déterminent si les ensembles de groupement dupliqués produisent chacun des lignes de sortie distinctes. Si ce paramètre n'est pas spécifié, ALL est utilisé.

grouping_expressions vous permettent d'effectuer des opérations de regroupement complexes. Vous pouvez utiliser des opérations de regroupement complexes pour effectuer une analyse qui nécessite une agrégation sur plusieurs ensembles de colonnes dans une seule requête.

L'élément grouping_expressions peut être une fonction quelconque, telle que SUM, AVG ou COUNT, exécutée sur les colonnes d'entrée.

Les expressions GROUP BY peuvent grouper les sorties par noms de colonne d'entrée qui n'apparaissent pas dans la sortie de l'instruction SELECT.

Toutes les expressions de sortie doivent être des fonctions d'agrégat ou des colonnes présentes dans la clause GROUP BY.

Vous pouvez utiliser une seule requête pour effectuer une analyse qui nécessite l'agrégation de plusieurs jeux de colonnes.

Athena prend en charge les agrégations complexes à l'aide de GROUPING SETS, CUBE et ROLLUP. GROUP BY GROUPING SETS spécifie plusieurs listes de colonnes à regrouper. GROUP BY CUBE génère tous les ensembles de regroupement possibles pour un ensemble de colonnes donné. GROUP BY ROLLUP génère tous les sous-totaux possibles pour un ensemble de colonnes donné. Les opérations de regroupement complexes ne prennent pas en charge le regroupement sur des expressions composées de colonnes d'entrée. Seuls les noms de colonnes sont autorisés.

Vous pouvez souvent utiliser UNION ALL pour obtenir les mêmes résultats que ces opérations GROUP BY, mais les requêtes qui utilisent GROUP BY ont l'avantage de lire les données une seule fois, tandis qu'UNION ALL lit les données sous-jacentes trois fois et peut générer des résultats incohérents lorsque la source de données est soumise à modification.

[HAVING condition]

Utilisé avec les fonctions d'agrégat et la clause GROUP BY. Détermine quels groupes sont sélectionnés, en éliminant ceux qui ne satisfont pas condition. Le filtrage se produit après le calcul des groupes et des agrégats.

[{ UNION | INTERSECT | EXCEPT } [ALL | DISTINCT] union_query]]

UNION, INTERSECT et EXCEPT combinent les résultats de plus d'une instruction SELECT en une seule requête. ALL et DISTINCT contrôlent l'unicité des lignes incluses dans le jeu de résultats final.

UNION combine les lignes résultant de la première requête avec les lignes résultant de la deuxième requête. Pour éliminer les doublons, UNION construit une table de hachage, qui consomme de la mémoire. Pour de meilleures performances, envisagez d'utiliser UNION ALL si votre requête ne nécessite pas l'élimination des doublons. Plusieurs clauses UNION sont traitées de gauche à droite, sauf si vous utilisez des parenthèses pour définir explicitement l'ordre de traitement.

INTERSECT renvoie uniquement les lignes qui sont présentes dans les résultats de la première et de la seconde requête.

EXCEPT renvoie les lignes des résultats de la première requête, en excluant les lignes trouvées par la seconde requête.

ALL entraîne l'inclusion de toutes les lignes, même si elles sont identiques.

DISTINCT fait en sorte que seules les lignes uniques soient incluses dans le jeu de résultats combinés.

[ORDER BY expression [ASC | DESC] [NULLS FIRST | NULLS LAST] [, ...]]

Trie un jeu de résultats par une ou expression de sortie.

Lorsque la clause contient plusieurs expressions, les résultats sont triés en fonction de la première expression. Ensuite, la seconde expression est appliquée aux lignes qui possèdent des valeurs correspondantes à partir de la première expression, et ainsi de suite.

Chaque expression peut spécifier les colonnes à partir de SELECT ou un nombre ordinal pour une colonne de sortie par son emplacement, à partir de un.

ORDER BY est évaluée comme la dernière étape après toute clause GROUP BY ou HAVING. ASC et DESC déterminent si les résultats sont triés dans l'ordre croissant ou décroissant.

L'ordre null par défaut est NULLS LAST, que l'ordre soit croissant ou décroissant.

[OFFSET count [ROW | ROWS]]

Utilisation de la clause OFFSET pour ignorer un certain nombre de lignes principales du jeu de résultats. Si la clause ORDER BY est présente, la clause OFFSET est évaluée sur un jeu de résultats triés, et le jeu reste trié après que les lignes ignorées aient été écartées. Si la requête n'a pas de clause ORDER BY, le choix des lignes à écarter est arbitraire. Si le nombre spécifié par OFFSET est égal ou dépasse la taille du jeu de résultat, le résultat final est vide.

LIMIT [count | ALL]

Limite le nombre de lignes dans le jeu de résultats à count. LIMIT ALL est identique à l'omission de la clause LIMIT. Si la requête n'a pas de clause ORDER BY, les résultats sont arbitraires.

TABLESAMPLE [BERNOULLI | SYSTEM] (pourcentage)

Opérateur facultatif pour sélectionner les lignes d'une table à partir d'une méthode d'échantillonnage.

BERNOULLI sélectionne chaque ligne à inclure dans l'exemple de la table avec une probabilité de percentage. Tous les blocs physiques de la table sont analysés, et certaines lignes sont ignorées en fonction de la comparaison entre le percentage de l'échantillon et une valeur aléatoire calculée lors de l'exécution.

Avec SYSTEM, la table est divisée en segments logiques de données, et la table est échantillonnée au niveau de cette granularité.

Soit toutes les lignes d'un segment sont sélectionnées, soit le segment est ignoré en fonction de la comparaison entre l'échantillon percentage et une valeur aléatoire calculée lors de l'exécution. L'échantillonnage SYSTEM dépend du connecteur. Cette méthode ne garantit pas de probabilités d'échantillonnage indépendantes.

[UNNEST (array_or_map) [WITH ORDINALITY]]

Développe un tableau ou une carte dans une relation. Les tableaux sont développés en une seule colonne. Les cartes sont développées en deux colonnes (clé, valeur).

Vous pouvez utiliser UNNEST avec plusieurs arguments, qui sont développés en plusieurs colonnes avec autant de lignes que l'argument ayant la plus haute cardinalité.

Les autres colonnes sont complétées avec les valeurs NULL.

La clause WITH ORDINALITY ajoute une colonne « ordinality » à la fin.

UNNEST est généralement utilisé avec une clause JOIN et peut référencer les colonnes à partir des relations sur le côté gauche de la jointure JOIN.

Obtention des emplacements de fichiers pour les données source dans Simple Storage Service (Amazon S3)

Pour connaître l'emplacement du fichier Simple Storage Service (Amazon S3) pour les données d'une ligne de table, vous pouvez utiliser "$path" dans une requête SELECT, comme dans l'exemple suivant :

```
SELECT "$path" FROM "my_database"."my_table" WHERE year=2019;
```

Cela renvoie un résultat comme le suivant :

```
s3://DOC-EXAMPLE-BUCKET/datasets_mytable/year=2019/data_file1.json
```

Pour obtenir une liste unique et triée des chemins d'accès aux noms de fichiers S3 pour les données d'une table, vous pouvez utiliser `SELECT DISTINCT` et `ORDER BY`, comme dans l'exemple suivant.

```
SELECT DISTINCT "$path" AS data_source_file
FROM sampledb.elb_logs
ORDER By data_source_file ASC
```

Pour renvoyer uniquement les noms de fichiers sans le chemin d'accès, vous pouvez passer `"$path"` comme paramètre à une fonction `regexp_extract`, comme dans l'exemple suivant.

```
SELECT DISTINCT regexp_extract("$path", '[^/]+$') AS data_source_file
FROM sampledb.elb_logs
ORDER By data_source_file ASC
```

Pour renvoyer les données d'un fichier spécifique, spécifiez le fichier dans la clause `WHERE`, comme dans l'exemple suivant.

```
SELECT *,"$path" FROM my_database.my_table WHERE "$path" = 's3://DOC-EXAMPLE-BUCKET/
my_table/my_partition/file-01.csv'
```

Pour plus d'informations et d'exemples, consultez l'article Comment voir le fichier source Simple Storage Service (Amazon S3) pour une ligne dans une table Athena ? du Centre de connaissances.

> **Note**
>
> Dans Athena, les colonnes de métadonnées masquées `$bucket`, `$file_modified_time`, `$file_size` et `$partition` ne sont pas prises en charge pour les vues.

Échappement de guillemets simples

Pour échapper un guillemet simple, faites-le précéder d'un autre guillemet simple, comme dans l'exemple suivant. Ne confondez pas ceci avec un guillemet double.

```
Select 'O''Reilly'
```

Résultats

```
O'Reilly
```

Ressources supplémentaires

Pour plus d'informations sur l'utilisation des instructions SELECT dans Athena, consultez les ressources suivantes.

Pour plus d'informations à ce sujet	Voir ce qui suit
Exécution de requêtes dans Athena	Exécution de requêtes SQL à l'aide d'Amazon Athena
Utilisation de SELECT pour créer une table	Création d'une table à partir des résultats des requêtes (CTAS)
Insertion de données à partir d'une requête SELECT dans une autre table	INSERT INTO
Utilisation de fonctions intégrées dans les instructions SELECT	Fonctions dans Amazon Athena
Utilisation de fonctions définies par l'utilisateur dans les instructions SELECT	Interrogation avec des fonctions définies par l'utilisateur
Interrogation des métadonnées d'un catalogue de données	Interrogation du AWS Glue Data Catalog

INSERT INTO

Insère de nouvelles lignes dans une table de destination en fonction d'une instruction de requête SELECT qui s'exécute sur une table source, ou en fonction d'un ensemble de VALUES fourni dans le cadre de l'instruction. Lorsque la table source est basée sur des données sous-jacentes dans un format tel que CSV ou JSON, et que la table de destination est basée sur un autre format, comme Parquet ou ORC, vous pouvez utiliser les requêtes INSERT INTO pour transformer les données sélectionnées au format de la table de destination.

Considérations et restrictions

Tenez compte des points suivants lorsque vous utilisez des requêtes INSERT avec Athena.

- Lors de l'exécution d'une requête `INSERT` sur une table avec des données sous-jacentes chiffrées dans Simple Storage Service (Amazon S3), les fichiers de sortie écrit par la requête `INSERT` ne sont pas chiffrés par défaut. Nous vous recommandons de chiffrer les résultats de la requête `INSERT` si vous les insérez dans des tables avec des données chiffrées.

  Pour plus d'informations sur le chiffrement des résultats de requête à l'aide de la console, consultez Chiffrement des résultats des requêtes Athena stockées dans Simple Storage Service (Amazon S3). Pour activer le chiffrement à l'aide de l'API AWS CLI ou Athena, utilisez les `EncryptionConfiguration` propriétés de l'StartQueryExecutionaction pour spécifier les options de chiffrement Amazon S3 en fonction de vos besoins.
- Pour les instructions `INSERT INTO`, le paramètre de propriétaire du compartiment attendu ne s'applique pas à l'emplacement de la table de destination dans Simple Storage Service (Amazon S3). Le paramètre de propriétaire du compartiment attendu s'applique uniquement à l'emplacement de sortie Simple Storage Service (Amazon S3) que vous spécifiez pour les résultats de la requête Athena. Pour plus d'informations, consultez Spécification d'un emplacement de résultats de requête à l'aide de la console Athena.
- Pour les instructions `INSERT INTO` conformes à la norme ACID, voir la rubrique `INSERT INTO` de Mise à jour des données de la table Iceberg.

Formats pris en charge et SerDes

Vous pouvez exécuter une `INSERT` requête sur des tables créées à partir de données aux formats suivants et SerDes.

Format de données	SerDe
Avro	org.apache.hadoop.hive.serde2.avro. AvroSerDe
Ion	com.amazon.ionhiveserde. IonHiveSerDe
JSON	org.apache.hive.hcatalog.data. JsonSerDe
ORC	org.apache.hadoop.hive.ql.io.orc. OrcSerde
Parquet	org.apache.hadoop.hive.ql.io.parquet.serde. ParquetHiveSerDe

Format de données	SerDe
Fichier texte	org.apache.hadoop.hive.serde2.lazy. LazySimpleSerDe **Note** Les fichiers CSV, TSV et avec séparateur personnalisé sont pris en charge.

Tables compartimentées non prises en charge

`INSERT INTO` n'est pas pris en charge sur les tables compartimentées. Pour plus d'informations, consultez Partitionnement et compartimentation dans Athena.

Requêtes fédérées non prises en charge

`INSERT INTO` n'est pas supporté pour les requêtes fédérées. Si vous tentez de le faire, le message d'erreur suivant peut s'afficher : This operation is currently not supported for external catalogs (Cette opération n'est actuellement pas prise en charge pour les catalogues externes). Pour plus d'informations sur les requêtes fédérées, consultez Utilisation de la requête fédérée d'Amazon Athena.

Partitioning

Tenez compte des points de cette section lorsque vous utilisez le partitionnement avec des requêtes `INSERT INTO` ou `CREATE TABLE AS SELECT`.

Limites

L'instruction `INSERT INTO` prend en charge l'écriture de 100 partitions au maximum dans la table de destination. Si vous exécutez la clause `SELECT` sur une table avec plus de 100 partitions, la requête échoue sauf si la requête `SELECT` est limitée à 100 partitions ou moins.

Pour plus d'informations sur le contournement de cette limitation, consultez Utilisation de CTAS et de INSERT INTO pour contourner la limite de 100 partitions.

Ordre des colonnes

Les instructions `INSERT INTO` ou `CREATE TABLE AS SELECT` s'attendent à ce que la colonne partitionnée soit la dernière colonne de la liste des colonnes projetées dans une instruction `SELECT`.

Si la table source n'est pas partitionnée, ou si elle est partitionnée sur différentes colonnes par rapport à la table de destination, les requêtes comme `INSERT INTO` *`destination_table`* `SELECT * FROM` *`source_table`* considèrent les valeurs de la dernière colonne de la table source comme des valeurs pour une colonne de partition dans la table de destination. Gardez ceci à l'esprit lorsque vous essayez de créer une table partitionnée à partir d'une table non partitionnée.

Ressources

Pour plus d'informations sur l'utilisation de `INSERT INTO` avec le partitionnement, consultez les ressources suivantes.

- Pour insérer des données partitionnées dans une table partitionnée, voir Utilisation de CTAS et de INSERT INTO pour contourner la limite de 100 partitions.
- Pour insérer des données non partitionnées dans une table partitionnée, voir Utilisation de CTAS et INSERT INTO pour ETL et l'analyse des données.

Fichiers écrits dans Simple Storage Service (Amazon S3)

Athena écrit des fichiers dans des emplacements de données source dans Simple Storage Service (Amazon S3) comme résultat de la commande `INSERT`. Chaque opération `INSERT` crée un nouveau fichier, plutôt que d'ajouter à un fichier existant. L'emplacement des fichiers dépend de la structure de la table et de la requête `SELECT`, si elle est présente. Athena génère un fichier manifeste de données pour chaque requête `INSERT`. Le manifeste suit les fichiers écrits par la requête. Il est enregistré dans l'emplacement des résultats de requête Athena dans Simple Storage Service (Amazon S3). Pour plus d'informations, consultez Identification des fichiers de sortie de requête.

Évitez les mises à jour hautement transactionnelles

Lorsque vous ajoutez `INSERT INTO` des lignes à une table dans Amazon S3, Athena ne réécrit ni ne modifie les fichiers existants. Au lieu de cela, il écrit les lignes sous la forme d'un ou de plusieurs nouveaux fichiers. Étant donné que les tables contenant de nombreux petits fichiers réduisent les performances des requêtes et que les opérations d'écriture `PutObject` et de lecture `GetObject` entraînent des coûts plus élevés pour Amazon S3, considérez les options suivantes lors de l'utilisation `INSERT INTO` :

- Exécutez `INSERT INTO` des opérations moins fréquemment sur des lots de lignes plus importants.
- Pour les gros volumes d'ingestion de données, pensez à utiliser un service tel qu'Amazon Data Firehose.

- Évitez `INSERT INTO` complètement de l'utiliser. Accumulez plutôt les lignes dans des fichiers plus volumineux et chargez-les directement sur Amazon S3, où Athena pourra les interroger.

Localisation de fichiers orphelins

Si une `INSERT INTO` instruction `CTAS` or échoue, les données orphelines peuvent être laissées à l'emplacement des données et peuvent être lues dans les requêtes suivantes. Pour localiser les fichiers orphelins en vue d'une inspection ou d'une suppression, vous pouvez utiliser le fichier manifeste de données fourni par Athena pour suivre la liste des fichiers à écrire. Pour plus d'informations, consultez Identification des fichiers de sortie de requête et DataManifestLocation.

INSERT INTO...SELECT

Spécifie la requête à exécuter sur une table, `source_table`, qui détermine les lignes à insérer dans une deuxième table, `destination_table`. Si la requête `SELECT` spécifie des colonnes de la `source_table`, les colonnes doivent correspondre exactement à celles de la `destination_table`.

Pour plus d'informations sur les requêtes `SELECT`, consultez SELECT.

Résumé

```
INSERT INTO destination_table
SELECT select_query
FROM source_table_or_view
```

Exemples

Sélectionner toutes les lignes de la table `vancouver_pageviews` et insérez-les dans la table `canada_pageviews` :

```
INSERT INTO canada_pageviews
SELECT *
FROM vancouver_pageviews;
```

Sélectionner uniquement les lignes de la table `vancouver_pageviews` où la colonne `date` a une valeur comprise entre `2019-07-01` et `2019-07-31`, puis les insérer dans `canada_july_pageviews` :

```
INSERT INTO canada_july_pageviews
SELECT *
FROM vancouver_pageviews
WHERE date
    BETWEEN date '2019-07-01'
        AND '2019-07-31';
```

Sélectionner les valeurs des colonnes `city` et `state` de la table `cities_world` uniquement à partir des lignes avec la valeur `usa` dans la colonne `country`, puis les insérer dans les colonnes `city` et `state` de la table `cities_usa` :

```
INSERT INTO cities_usa (city,state)
SELECT city,state
FROM cities_world
    WHERE country='usa'
```

INSERT INTO...VALUES

Insère des lignes dans une table existante en spécifiant des colonnes et des valeurs. Les colonnes spécifiées et les types de données associés doivent correspondre précisément aux colonnes et aux types de données de la table de destination.

> **Important**
>
> Nous vous déconseillons d'insérer des lignes à l'aide de `VALUES`, car Athena génère des fichiers pour chaque opération `INSERT`. Cela peut entraîner la création de nombreux petits fichiers et dégrader les performances de requête de la table. Pour identifier les fichiers créés par une requête `INSERT`, examinez le fichier manifeste de données. Pour plus d'informations, consultez Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie.

Résumé

```
INSERT INTO destination_table [(col1,col2,...)]
VALUES (col1value,col2value,...)[,
       (col1value,col2value,...)][,
       ...]
```

Exemples

Dans les exemples suivants, la table cities comporte quatre colonnes : `id`, `city`, `state` et `state_motto`. La colonne `id` est de type `INT` et toutes les autres colonnes sont de type `VARCHAR`.

Insérer une seule ligne dans la table `cities`, avec toutes les valeurs de colonne spécifiées :

```
INSERT INTO cities
VALUES (1,'Lansing','MI','Si quaeris peninsulam amoenam circumspice')
```

Insérer deux lignes dans la table `cities` :

```
INSERT INTO cities
VALUES (1,'Lansing','MI','Si quaeris peninsulam amoenam circumspice'),
       (3,'Boise','ID','Esto perpetua')
```

DELETE

Permet de supprimer des lignes d'une table Apache Iceberg. `DELETE` est transactionnel et n'est pris en charge que pour les tables Apache Iceberg.

Résumé

Pour supprimer les lignes d'une table Iceberg, utilisez la syntaxe suivante.

```
DELETE FROM [db_name.]table_name [WHERE predicate]
```

Pour plus d'informations et d'exemples, voir la rubrique `DELETE` de Mise à jour des données de la table Iceberg.

MISE A JOUR

Permet de mettre à jour les lignes d'une table Apache Iceberg. `UPDATE` est transactionnel et n'est pris en charge que pour les tables Apache Iceberg.

Résumé

Pour mettre à jour les lignes d'une table Iceberg, utilisez la syntaxe suivante.

```
UPDATE [db_name.]table_name SET xx=yy[,...] [WHERE predicate]
```

Pour plus d'informations et d'exemples, voir la rubrique UPDATE de Mise à jour des données de la table Iceberg.

MERGE INTO

Mise à jour, suppression ou insertion conditionnelle de lignes dans une table Apache Iceberg. Une seule instruction peut combiner des actions de mise à jour, de suppression et d'insertion.

> **Note**
>
> MERGE INTO est transactionnel et n'est pris en charge que pour les tables Apache Iceberg dans la version 3 du moteur Athena.

Résumé

Pour mettre à jour, supprimer ou insérer de manière conditionnelle des lignes d'une table Iceberg, utilisez la syntaxe suivante.

```
MERGE INTO target_table [ [ AS ]  target_alias ]
USING { source_table | query } [ [ AS ] source_alias ]
ON search_condition
when_clause [...]
```

La clause *when_clause* est l'une des suivantes :

```
WHEN MATCHED [ AND condition ]
    THEN DELETE
```

```
WHEN MATCHED [ AND condition ]
    THEN UPDATE SET ( column = expression [, ...] )
```

```
WHEN NOT MATCHED [ AND condition ]
    THEN INSERT (column_name[, column_name ...]) VALUES (expression, ...)
```

MERGE prend en charge un nombre arbitraire de clauses WHEN avec des conditions MATCHED différentes. Les clauses de condition exécutent l'opération DELETE, UPDATE ou INSERT dans la première clause WHEN sélectionnée par l'état MATCHED et la condition de correspondance.

Pour chaque ligne source, les clauses WHEN sont traitées dans l'ordre. Seule la première clause WHEN correspondante est exécutée. Les clauses suivantes sont ignorées. Une erreur utilisateur est signalée lorsqu'une seule ligne de la table cible correspond à plus d'une ligne source.

Si une ligne source ne correspond à aucune clause WHEN et qu'il n'y a pas de clause WHEN NOT MATCHED, la ligne source est ignorée.

Dans les clauses WHEN qui comportent des opérations UPDATE, les expressions de valeur de colonne peuvent renvoyer à n'importe quel champ de la cible ou de la source. Dans le cas de NOT MATCHED, les expressions INSERT peuvent renvoyer à n'importe quel champ de la source.

Exemple

L'exemple suivant fusionne les lignes de la deuxième table dans la première table si elles n'existent pas dans la première table. Notez que les colonnes répertoriées dans la clause VALUES doivent être préfixées par l'alias de la table source. Les colonnes cibles répertoriées dans la clause INSERT ne doivent pas être préfixées.

```
MERGE INTO iceberg_table_sample as ice1
USING iceberg2_table_sample as ice2
ON ice1.col1 = ice2.col1
WHEN NOT MATCHED
THEN INSERT (col1)
      VALUES (ice2.col1)
```

Pour obtenir plus d'exemples MERGE INTO, consultez Mise à jour des données de la table Iceberg.

OPTIMIZE

Optimise les lignes d'une table Apache Iceberg en réécrivant les fichiers de données dans une disposition plus optimisée en fonction de leur taille et du nombre de fichiers de suppression associés.

> **Note**
>
> OPTIMIZE est transactionnel et n'est pris en charge que pour les tables Apache Iceberg.

Syntaxe

Le résumé syntaxique suivant montre comment optimiser la mise en page des données pour une table Iceberg.

```
OPTIMIZE [db_name.]table_name REWRITE DATA USING BIN_PACK
  [WHERE predicate]
```

> **Note**
>
> Seules les colonnes de partition sont autorisées dans le *prédicat* de WHERE clause. La spécification d'une colonne non partitionnée entraînera l'échec de la requête.

L'action de compactage est facturée en fonction de la quantité de données analysées pendant le processus de réécriture. L'action REWRITE DATA utilise des prédicats pour sélectionner les fichiers contenant des lignes correspondantes. Si une ligne du fichier correspond au prédicat, le fichier est sélectionné pour optimisation. Ainsi, pour contrôler le nombre de fichiers affectés par l'opération de compactage, vous pouvez spécifier une clause WHERE.

Configuration des propriétés de compactage

Pour contrôler la taille des fichiers à sélectionner pour le compactage et la taille du fichier résultant après le compactage, vous pouvez utiliser les paramètres de propriété de table. Vous pouvez utiliser l'instruction ALTER TABLE SET PROPERTIES pour configurer les propriétés de la table associée.

Ressources supplémentaires

Optimisation des tables Iceberg

VACUUM

L'instruction VACUUM effectue la maintenance des tables d'Apache Iceberg en supprimant les fichiers de données devenus inutiles.

> **Note**
>
> VACUUM est transactionnel et n'est pris en charge que pour les tables Apache Iceberg dans la version 3 du moteur Athena.

Il est recommandé d'exécuter l'instruction VACUUM sur les tables Iceberg pour supprimer les fichiers de données qui ne sont plus pertinents et pour réduire la taille des métadonnées et la consommation

de stockage. Notez que, dans la mesure où l'instruction VACUUM envoie des appels d'API à Amazon S3, des frais s'appliquent pour les demandes associées à Amazon S3.

> **Warning**
>
> Si vous exécutez une opération d'expiration des instantanés, vous ne pouvez plus parcourir le temps vers les instantanés expirés.

Résumé

Pour supprimer les fichiers de données qui ne sont plus nécessaires pour une table Iceberg, utilisez la syntaxe suivante.

```
VACUUM [database_name.]target_table
```

Pour exécuter une VACUUM opération sur une table dont le nom commence par un trait de soulignement (par exemple,_mytable), placez le nom de la table en backticks, comme dans l'exemple suivant. Si vous préfixez le nom de la table par un nom de base de données, ne le mettez pas entre crochets. Notez que les guillemets ne remplaceront pas les backticks.

Ce comportement est propre àVACUUM. Les INSERT INTO instructions CREATE and ne nécessitent pas de cocher la case arrière pour les noms de table commençant par un trait de soulignement.

```
VACUUM `_mytable`
VACUUM my_database.`_mytable`
```

Notez également que les VACUUM données Iceberg devraient se trouver dans un dossier Amazon S3 plutôt que dans un compartiment Amazon S3. Par exemple, si vos données Iceberg se trouvent às3://DOC-EXAMPLE-BUCKET/au lieu des3://DOC-EXAMPLE-BUCKET/myicebergfolder/, l'VACUUMinstruction échoue avec le message d'erreur GENERIC_INTERNAL_ERROR : Path missing in file system location :. s3://DOC-EXAMPLE-BUCKET

Opérations effectuées

VACUUM effectue les opérations suivantes :

- Supprime les instantanés qui sont plus anciens que la période spécifiée par la propriété vacuum_max_snapshot_age_seconds de la table. Par défaut, cette propriété est définie sur 432 000 secondes (5 jours).

- Supprime les instantanés qui ne sont pas dans la période de rétention et qui dépassent le nombre spécifié par la propriété `vacuum_min_snapshots_to_keep` de la table. La valeur par défaut est 1.

  Vous pouvez spécifier ces propriétés de table dans votre instruction `CREATE TABLE`. Une fois la table créée, vous pouvez utiliser l'instruction ALTER TABLE SET PROPERTIES pour la mettre à jour.
- Supprime tous les fichiers de métadonnées et de données qui sont inaccessibles à la suite de la suppression de l'instantané. Vous pouvez configurer le nombre d'anciens fichiers de métadonnées à conserver en définissant la propriété de table `vacuum_max_metadata_files_to_keep`. La valeur par défaut est 100.
- Supprime les fichiers orphelins qui sont plus anciens que le temps spécifié dans la propriété `vacuum_max_snapshot_age_seconds` de la table. Les fichiers orphelins sont des fichiers dans le répertoire de données de la table qui ne font pas partie de l'état de la table.

Pour plus d'informations sur la création et la gestion de tables Apache Iceberg dans Athena, voir Création de tables Iceberg et Gestion des tables Iceberg.

Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena

L'instruction `EXPLAIN` montre le plan d'exécution logique ou distribué d'une instruction SQL spécifiée, ou valide l'instruction SQL. Vous pouvez produire les résultats au format texte ou dans un format de données pour les intégrer dans un graphique.

> **Note**
>
> Vous pouvez afficher des représentations graphiques des plans logiques et distribués pour vos requêtes dans la console Athena sans utiliser la syntaxe `EXPLAIN`. Pour plus d'informations, consultez Affichage des plans d'exécution pour les requêtes SQL.

L'instruction `EXPLAIN ANALYZE` montre à la fois le plan d'exécution distribué d'une instruction SQL spécifiée et le coût de calcul de chaque opération dans une requête SQL. Vous pouvez afficher les résultats au format texte ou JSON.

Considérations et restrictions

Les instructions `EXPLAIN` et `EXPLAIN ANALYZE` sur Athena ont les limitations suivantes.

- Comme les requêtes `EXPLAIN` n'analysent pas de données, Athena ne les facture pas. Cependant, étant donné que les requêtes `EXPLAIN` effectuent des appels vers AWS Glue pour récupérer les métadonnées des tables, vous pouvez encourir des frais de la part de Glue si les appels dépassent la limite de l'offre gratuite de Glue.
- Étant donné que les requêtes `EXPLAIN ANALYZE` sont exécutées, elles analysent les données et Athena facture la quantité de données analysées.
- Les informations de filtrage des lignes ou des cellules définies dans Lake Formation et les informations sur les statistiques des requêtes n'apparaissent pas dans la sortie de `EXPLAIN` et `EXPLAIN ANALYZE`.

Syntaxe EXPLAIN

```
EXPLAIN [ ( option [, ...]) ] statement
```

L'*option* peut être l'une des suivantes :

```
FORMAT { TEXT | GRAPHVIZ | JSON }
TYPE { LOGICAL | DISTRIBUTED | VALIDATE | IO }
```

Si l'option `FORMAT` n'est pas spécifiée, la sortie se fait par défaut au format `TEXT`. Le type `IO` fournit des informations sur les tables et les schémas que la requête lit. `IO` n'est pris en charge que dans la version 2 du moteur Athena et ne peut être renvoyé qu'au format JSON.

Syntaxe EXPLAIN ANALYZE

En plus de la sortie incluse dans `EXPLAIN`, la sortie `EXPLAIN ANALYZE` comprend également des statistiques d'exécution pour la requête spécifiée, telles que l'utilisation du CPU, le nombre de lignes en entrée et le nombre de lignes en sortie.

```
EXPLAIN ANALYZE [ ( option [, ...]) ] statement
```

L'*option* peut être l'une des suivantes :

```
FORMAT { TEXT | JSON }
```

Si l'option `FORMAT` n'est pas spécifiée, la sortie se fait par défaut au format `TEXT`. Parce que toutes les requêtes pour `EXPLAIN ANALYZE` sont `DISTRIBUTED`, l'option `TYPE` n'est pas disponible pour `EXPLAIN ANALYZE`.

L'instruction peut être l'une des suivantes :

```
SELECT
CREATE TABLE AS SELECT
INSERT
UNLOAD
```

Exemples EXPLAIN

Les exemples suivants pour EXPLAIN vont du plus simple au plus complexe.

Exemple 1 EXPLAIN : utilisez l'instruction EXPLAIN pour afficher un plan de requête au format texte

Dans l'exemple suivant, EXPLAIN affiche le plan d'exécution d'une requête SELECT sur les journaux Elastic Load Balancing. Le format est défini par défaut sur la sortie de texte.

```
EXPLAIN
SELECT
   request_timestamp,
   elb_name,
   request_ip
FROM sampledb.elb_logs;
```

Résultats

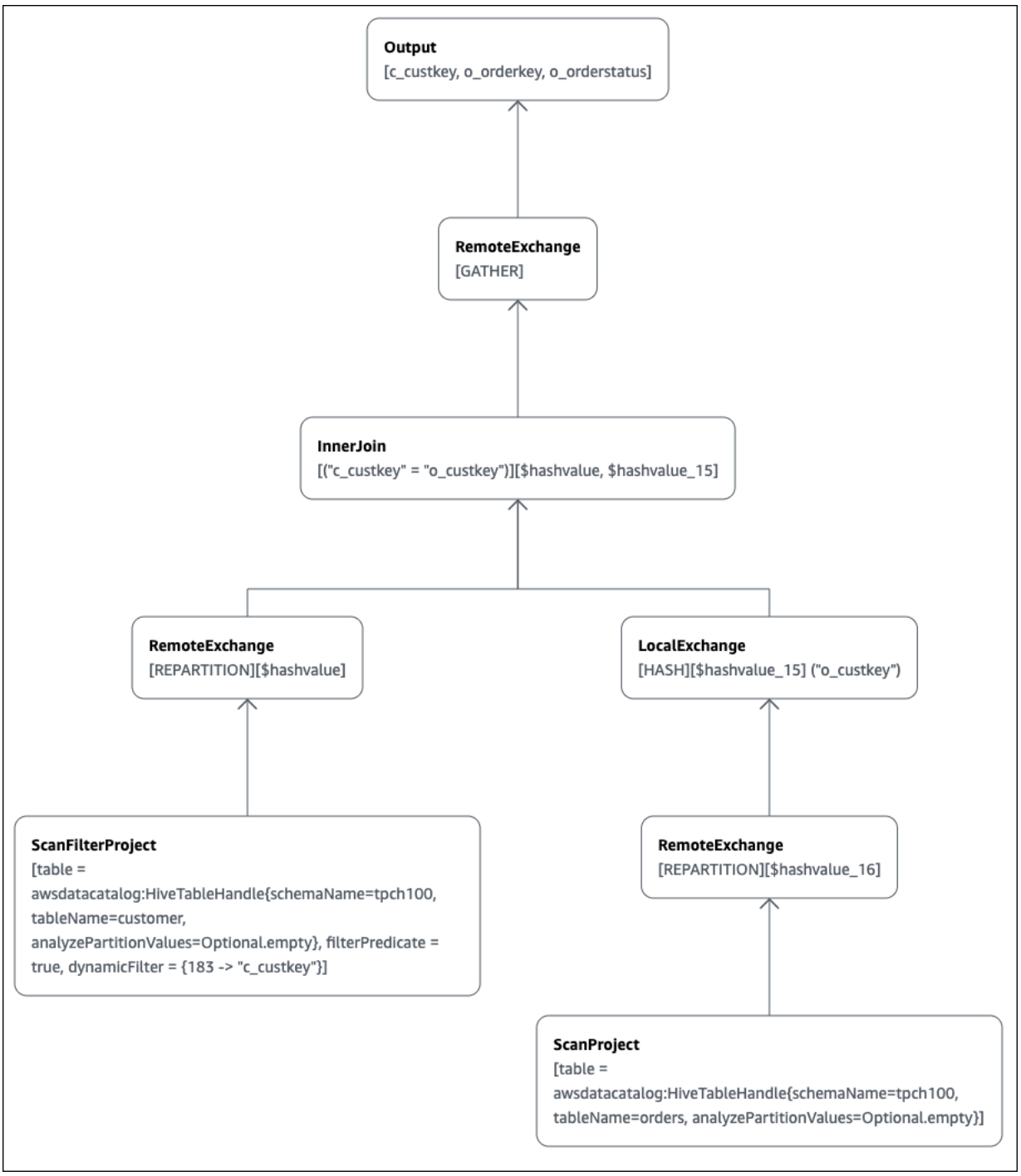
```
- Output[request_timestamp, elb_name, request_ip] => [[request_timestamp, elb_name,
 request_ip]]
    - RemoteExchange[GATHER] => [[request_timestamp, elb_name, request_ip]]
        - TableScan[awsdatacatalog:HiveTableHandle{schemaName=sampledb,
 tableName=elb_logs,
analyzePartitionValues=Optional.empty}] => [[request_timestamp, elb_name, request_ip]]
                LAYOUT: sampledb.elb_logs
                request_ip := request_ip:string:2:REGULAR
                request_timestamp := request_timestamp:string:0:REGULAR
                elb_name := elb_name:string:1:REGULAR
```

Exemple 2 EXPLAIN : représentez graphiquement un plan de requête

Vous pouvez utiliser la console Athena pour représenter graphiquement un plan de requête pour vous. Saisissez une instruction SELECT comme la suivante dans l'éditeur de requêtes Athena, puis choisissez EXPLAIN.

```
SELECT
      c.c_custkey,
      o.o_orderkey,
      o.o_orderstatus
   FROM tpch100.customer c
   JOIN tpch100.orders o
       ON c.c_custkey = o.o_custkey
```

La page EXPLAIN de l'éditeur de requêtes Athena s'ouvre et affiche un plan distribué et un plan logique pour la requête. Le graphique suivant montre le plan logique de l'exemple.



Important

Actuellement, certains filtres de partition peuvent ne pas être visibles dans l'arborescence des opérateurs imbriqués même si Athena les applique à votre requête. Pour vérifier l'effet de tels filtres, exécutez `EXPLAIN` ou `EXPLAIN ANALYZE` sur votre requête et affichez les résultats.

Pour plus d'informations sur l'utilisation des fonctionnalités de graphique du plan de requête dans la console Athena, consultez Affichage des plans d'exécution pour les requêtes SQL.

Exemple 3 EXPLAIN : utilisez l'instruction EXPLAIN pour vérifier le nettoyage des partitions

Lorsque vous utilisez un prédicat de filtrage sur une clé partitionnée pour effectuer une requête sur une table partitionnée, le moteur de requête applique le prédicat à la clé partitionnée afin de réduire la quantité de données lues.

L'exemple suivant utilise une requête `EXPLAIN` pour vérifier l'élagage des partitions pour une requête `SELECT` sur une table partitionnée. Tout d'abord, une instruction `CREATE TABLE` crée la table `tpch100.orders_partitioned`. La table est partitionnée sur la colonne `o_orderdate`.

```
CREATE TABLE `tpch100.orders_partitioned`(
  `o_orderkey` int,
  `o_custkey` int,
  `o_orderstatus` string,
  `o_totalprice` double,
  `o_orderpriority` string,
  `o_clerk` string,
  `o_shippriority` int,
  `o_comment` string)
PARTITIONED BY (
  `o_orderdate` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/<your_directory_path>/'
```

La table `tpch100.orders_partitioned` a plusieurs partitions sur `o_orderdate`, comme le montre la commande `SHOW PARTITIONS`.

```
SHOW PARTITIONS tpch100.orders_partitioned;

o_orderdate=1994
o_orderdate=2015
o_orderdate=1998
o_orderdate=1995
```

```
o_orderdate=1993
o_orderdate=1997
o_orderdate=1992
o_orderdate=1996
```

La requête EXPLAIN suivante vérifie l'élagage des partitions sur l'instruction SELECT spécifiée.

```
EXPLAIN
SELECT
   o_orderkey,
   o_custkey,
   o_orderdate
FROM tpch100.orders_partitioned
WHERE o_orderdate = '1995'
```

Résultats

```
Query Plan
- Output[o_orderkey, o_custkey, o_orderdate] => [[o_orderkey, o_custkey, o_orderdate]]
    - RemoteExchange[GATHER] => [[o_orderkey, o_custkey, o_orderdate]]
        - TableScan[awsdatacatalog:HiveTableHandle{schemaName=tpch100,
 tableName=orders_partitioned,
analyzePartitionValues=Optional.empty}] => [[o_orderkey, o_custkey, o_orderdate]]
                LAYOUT: tpch100.orders_partitioned
                o_orderdate := o_orderdate:string:-1:PARTITION_KEY
                    :: [[1995]]
                o_custkey := o_custkey:int:1:REGULAR
                o_orderkey := o_orderkey:int:0:REGULAR
```

Le texte en gras dans le résultat montre que le prédicat o_orderdate = '1995' a été appliqué sur la PARTITION_KEY.

Exemple 4 EXPLAIN : utilisez une requête EXPLAIN pour vérifier l'ordre et le type de jointure

La requête EXPLAIN suivante vérifie l'ordre et le type de jointure de l'instruction SELECT. Utilisez une requête comme celle-ci pour examiner l'utilisation de la mémoire de la requête afin de réduire les risques d'erreur EXCEEDED_LOCAL_MEMORY_LIMIT.

```
EXPLAIN (TYPE DISTRIBUTED)
   SELECT
      c.c_custkey,
```

```
      o.o_orderkey,
      o.o_orderstatus
   FROM tpch100.customer c
   JOIN tpch100.orders o
       ON c.c_custkey = o.o_custkey
   WHERE c.c_custkey = 123
```

Résultats

```
Query Plan
Fragment 0 [SINGLE]
    Output layout: [c_custkey, o_orderkey, o_orderstatus]
    Output partitioning: SINGLE []
    Stage Execution Strategy: UNGROUPED_EXECUTION
    - Output[c_custkey, o_orderkey, o_orderstatus] => [[c_custkey, o_orderkey,
 o_orderstatus]]
        - RemoteSource[1] => [[c_custkey, o_orderstatus, o_orderkey]]

Fragment 1 [SOURCE]
    Output layout: [c_custkey, o_orderstatus, o_orderkey]
    Output partitioning: SINGLE []
    Stage Execution Strategy: UNGROUPED_EXECUTION
    - CrossJoin => [[c_custkey, o_orderstatus, o_orderkey]]
            Distribution: REPLICATED
        - ScanFilter[table = awsdatacatalog:HiveTableHandle{schemaName=tpch100,
tableName=customer, analyzePartitionValues=Optional.empty}, grouped = false,
filterPredicate = ("c_custkey" = 123)] => [[c_custkey]]
                LAYOUT: tpch100.customer
                c_custkey := c_custkey:int:0:REGULAR
        - LocalExchange[SINGLE] () => [[o_orderstatus, o_orderkey]]
            - RemoteSource[2] => [[o_orderstatus, o_orderkey]]

Fragment 2 [SOURCE]
    Output layout: [o_orderstatus, o_orderkey]
    Output partitioning: BROADCAST []
    Stage Execution Strategy: UNGROUPED_EXECUTION
    - ScanFilterProject[table = awsdatacatalog:HiveTableHandle{schemaName=tpch100,
tableName=orders, analyzePartitionValues=Optional.empty}, grouped = false,
filterPredicate = ("o_custkey" = 123)] => [[o_orderstatus, o_orderkey]]
            LAYOUT: tpch100.orders
            o_orderstatus := o_orderstatus:string:2:REGULAR
            o_custkey := o_custkey:int:1:REGULAR
            o_orderkey := o_orderkey:int:0:REGULAR
```

La requête donnée en exemple a été optimisée en une jointure croisée pour une meilleure performance. Les résultats montrent que `tpch100.orders` sera distribué comme le type de distribution `BROADCAST`. Cela implique que la table `tpch100.orders` sera distribuée à tous les nœuds qui effectuent l'opération de jointure. Le type de distribution `BROADCAST` exigera que tous les résultats filtrés de la table `tpch100.orders` tiennent dans la mémoire de chaque nœud qui effectue l'opération de jointure.

Cependant, la table `tpch100.customer` est plus petite que `tpch100.orders`. Comme `tpch100.customer` nécessite moins de mémoire, vous pouvez réécrire la requête en `BROADCAST` `tpch100.customer` au lieu de `tpch100.orders`. Cela réduit le risque que la requête reçoive une erreur `EXCEEDED_LOCAL_MEMORY_LIMIT`. Cette politique suppose les points suivants :

- La `tpch100.customer.c_custkey` est unique dans la table `tpch100.customer`.
- Il existe une relation one-to-many cartographique entre `tpch100.customer` et`tpch100.orders`.

L'exemple suivant illustre la requête réécrite.

```
SELECT
    c.c_custkey,
    o.o_orderkey,
    o.o_orderstatus
FROM tpch100.orders o
JOIN tpch100.customer c -- the filtered results of tpch100.customer are distributed to
 all nodes.
    ON c.c_custkey = o.o_custkey
WHERE c.c_custkey = 123
```

Exemple 5 EXPLAIN : utilisez une requête EXPLAIN pour supprimer les prédicats qui n'ont aucun effet

Vous pouvez utiliser une requête `EXPLAIN` pour vérifier l'efficacité du filtrage des prédicats. Vous pouvez utiliser les résultats pour supprimer les prédicats qui n'ont aucun effet, comme dans l'exemple suivant.

```
EXPLAIN
   SELECT
      c.c_name
   FROM tpch100.customer c
   WHERE c.c_custkey = CAST(RANDOM() * 1000 AS INT)
```

```
   AND c.c_custkey BETWEEN 1000 AND 2000
   AND c.c_custkey = 1500
```

Résultats

```
Query Plan
- Output[c_name] => [[c_name]]
    - RemoteExchange[GATHER] => [[c_name]]
        - ScanFilterProject[table =
awsdatacatalog:HiveTableHandle{schemaName=tpch100,
tableName=customer, analyzePartitionValues=Optional.empty},
filterPredicate = (("c_custkey" = 1500) AND ("c_custkey" =
CAST(("random"() * 1E3) AS int)))] => [[c_name]]
                LAYOUT: tpch100.customer
                c_custkey := c_custkey:int:0:REGULAR
                c_name := c_name:string:1:REGULAR
```

Le `filterPredicate` dans les résultats montre que l'optimiseur a fusionné les trois prédicats d'origine en deux prédicats et a changé leur ordre d'application.

```
filterPredicate = (("c_custkey" = 1500) AND ("c_custkey" = CAST(("random"() * 1E3) AS
 int)))
```

Comme les résultats montrent que le prédicat `AND c.c_custkey BETWEEN 1000 AND 2000` n'a aucun effet, vous pouvez supprimer ce prédicat sans modifier les résultats de la requête.

Pour plus d'informations sur les termes utilisés dans les résultats des requêtes `EXPLAIN`, voir Explication des résultats de l'instruction EXPLAIN d'Athena.

Exemples EXPLAIN ANALYZE

Les exemples suivants montrent des exemples de requêtes et de sorties `EXPLAIN ANALYZE`.

Exemple 1 EXPLAIN ANALYZE : utilisez EXPLAIN ANALYZE pour afficher un plan de requête et un coût de calcul au format texte

L'exemple suivant `EXPLAIN ANALYZE` montre le plan d'exécution et les coûts de calcul d'une `SELECT` requête sur les CloudFront journaux. Le format est défini par défaut sur la sortie de texte.

```
EXPLAIN ANALYZE SELECT FROM cloudfront_logs LIMIT 10
```

Résultats

```
 Fragment 1
     CPU: 24.60ms, Input: 10 rows (1.48kB); per task: std.dev.: 0.00, Output: 10 rows
 (1.48kB)
     Output layout: [date, time, location, bytes, requestip, method, host, uri, status,
 referrer,\
       os, browser, browserversion]
Limit[10] => [[date, time, location, bytes, requestip, method, host, uri, status,
 referrer, os,\
  browser, browserversion]]
             CPU: 1.00ms (0.03%), Output: 10 rows (1.48kB)
             Input avg.: 10.00 rows, Input std.dev.: 0.00%
LocalExchange[SINGLE] () => [[date, time, location, bytes, requestip, method, host,
 uri, status, referrer, os,\
 browser, browserversion]]
                 CPU: 0.00ns (0.00%), Output: 10 rows (1.48kB)
                 Input avg.: 0.63 rows, Input std.dev.: 387.30%
RemoteSource[2] => [[date, time, location, bytes, requestip, method, host, uri, status,
 referrer, os,\
  browser, browserversion]]
                     CPU: 1.00ms (0.03%), Output: 10 rows (1.48kB)
                     Input avg.: 0.63 rows, Input std.dev.: 387.30%

 Fragment 2
     CPU: 3.83s, Input: 998 rows (147.21kB); per task: std.dev.: 0.00, Output: 20 rows
 (2.95kB)
     Output layout: [date, time, location, bytes, requestip, method, host, uri, status,
 referrer, os,\
       browser, browserversion]
LimitPartial[10] => [[date, time, location, bytes, requestip, method, host, uri,
 status, referrer, os,\
  browser, browserversion]]
             CPU: 5.00ms (0.13%), Output: 20 rows (2.95kB)
             Input avg.: 166.33 rows, Input std.dev.: 141.42%
TableScan[awsdatacatalog:HiveTableHandle{schemaName=default, tableName=cloudfront_logs,
\
  analyzePartitionValues=Optional.empty},
grouped = false] => [[date, time, location, bytes, requestip, method, host, uri, st
                 CPU: 3.82s (99.82%), Output: 998 rows (147.21kB)
                 Input avg.: 166.33 rows, Input std.dev.: 141.42%
                 LAYOUT: default.cloudfront_logs
                 date := date:date:0:REGULAR
                 referrer := referrer:string:9:REGULAR
```

```
                os := os:string:10:REGULAR
                method := method:string:5:REGULAR
                bytes := bytes:int:3:REGULAR
                browser := browser:string:11:REGULAR
                host := host:string:6:REGULAR
                requestip := requestip:string:4:REGULAR
                location := location:string:2:REGULAR
                time := time:string:1:REGULAR
                uri := uri:string:7:REGULAR
                browserversion := browserversion:string:12:REGULAR
                status := status:int:8:REGULAR
```

Exemple 2 EXPLAIN ANALYZE : utilisez EXPLAIN ANALYZE pour afficher un plan de requête au format JSON

L'exemple suivant montre le plan d'exécution et les coûts de calcul d'une SELECT requête sur les CloudFront journaux. L'exemple spécifie JSON comme format de sortie.

```
EXPLAIN ANALYZE (FORMAT JSON) SELECT * FROM cloudfront_logs LIMIT 10
```

Résultats

```
{
    "fragments": [{
        "id": "1",

        "stageStats": {
            "totalCpuTime": "3.31ms",
            "inputRows": "10 rows",
            "inputDataSize": "1514B",
            "stdDevInputRows": "0.00",
            "outputRows": "10 rows",
            "outputDataSize": "1514B"
        },
        "outputLayout": "date, time, location, bytes, requestip, method, host,\
           uri, status, referrer, os, browser, browserversion",

        "logicalPlan": {
            "1": [{
                "name": "Limit",
                "identifier": "[10]",
                "outputs": ["date", "time", "location", "bytes", "requestip", "method",
 "host",\
```

```
                  "uri", "status", "referrer", "os", "browser", "browserversion"],
                "details": "",
                "distributedNodeStats": {
                    "nodeCpuTime": "0.00ns",
                    "nodeOutputRows": 10,
                    "nodeOutputDataSize": "1514B",
                    "operatorInputRowsStats": [{
                        "nodeInputRows": 10.0,
                        "nodeInputRowsStdDev": 0.0
                    }]
                },
                "children": [{
                    "name": "LocalExchange",
                    "identifier": "[SINGLE] ()",
                    "outputs": ["date", "time", "location", "bytes", "requestip",
"method", "host",\
                      "uri", "status", "referrer", "os", "browser", "browserversion"],
                    "details": "",
                    "distributedNodeStats": {
                        "nodeCpuTime": "0.00ns",
                        "nodeOutputRows": 10,
                        "nodeOutputDataSize": "1514B",
                        "operatorInputRowsStats": [{
                            "nodeInputRows": 0.625,
                            "nodeInputRowsStdDev": 387.2983346207417
                        }]
                    },
                    "children": [{
                        "name": "RemoteSource",
                        "identifier": "[2]",
                        "outputs": ["date", "time", "location", "bytes", "requestip",
"method", "host",\
                          "uri", "status", "referrer", "os", "browser",
"browserversion"],
                        "details": "",
                        "distributedNodeStats": {
                            "nodeCpuTime": "0.00ns",
                            "nodeOutputRows": 10,
                            "nodeOutputDataSize": "1514B",
                            "operatorInputRowsStats": [{
                                "nodeInputRows": 0.625,
                                "nodeInputRowsStdDev": 387.2983346207417
                            }]
                        },
```

```
                    "children": []
                }]
            }]
        }]
    }
}, {
    "id": "2",

    "stageStats": {
        "totalCpuTime": "1.62s",
        "inputRows": "500 rows",
        "inputDataSize": "75564B",
        "stdDevInputRows": "0.00",
        "outputRows": "10 rows",
        "outputDataSize": "1514B"
    },
    "outputLayout": "date, time, location, bytes, requestip, method, host, uri,
status,\
       referrer, os, browser, browserversion",

    "logicalPlan": {
        "1": [{
            "name": "LimitPartial",
            "identifier": "[10]",
            "outputs": ["date", "time", "location", "bytes", "requestip", "method",
"host", "uri",\
              "status", "referrer", "os", "browser", "browserversion"],
            "details": "",
            "distributedNodeStats": {
                "nodeCpuTime": "0.00ns",
                "nodeOutputRows": 10,
                "nodeOutputDataSize": "1514B",
                "operatorInputRowsStats": [{
                    "nodeInputRows": 83.33333333333333,
                    "nodeInputRowsStdDev": 223.60679774997897
                }]
            },
            "children": [{
                "name": "TableScan",
                "identifier": "[awsdatacatalog:HiveTableHandle{schemaName=default,\
                   tableName=cloudfront_logs,
analyzePartitionValues=Optional.empty},\
                   grouped = false]",
```

```
                    "outputs": ["date", "time", "location", "bytes", "requestip",
 "method", "host", "uri",\
                       "status", "referrer", "os", "browser", "browserversion"],
                    "details": "LAYOUT: default.cloudfront_logs\ndate :=
 date:date:0:REGULAR\nreferrer :=\
                       referrer: string:9:REGULAR\nos := os:string:10:REGULAR
\nmethod := method:string:5:\
                       REGULAR\nbytes := bytes:int:3:REGULAR\nbrowser :=
 browser:string:11:REGULAR\nhost :=\
                       host:string:6:REGULAR\nrequestip := requestip:string:4:REGULAR
\nlocation :=\
                       location:string:2:REGULAR\ntime := time:string:1: REGULAR
\nuri := uri:string:7:\
                       REGULAR\nbrowserversion := browserversion:string:12:REGULAR
\nstatus :=\
                       status:int:8:REGULAR\n",
                    "distributedNodeStats": {
                        "nodeCpuTime": "1.62s",
                        "nodeOutputRows": 500,
                        "nodeOutputDataSize": "75564B",
                        "operatorInputRowsStats": [{
                            "nodeInputRows": 83.33333333333333,
                            "nodeInputRowsStdDev": 223.60679774997897
                        }]
                    },
                    "children": []
                }]
            }]
        }
    }]
}
```

Ressources supplémentaires

Pour plus d'informations, consultez les ressources suivantes.

- Explication des résultats de l'instruction EXPLAIN d'Athena
- Affichage des plans d'exécution pour les requêtes SQL
- Affichage des statistiques et des détails d'exécution pour les requêtes terminées
- Documentation Trino `EXPLAIN`
- Documentation Trino `EXPLAIN ANALYZE`

- Optimisez les performances des requêtes fédérées à l'aide d'EXPLAIN et EXPLAIN ANALYZE dans Amazon Athena sur le blog AWS Big Data.

Explication des résultats de l'instruction EXPLAIN d'Athena

Cette rubrique fournit un bref guide des termes opérationnels utilisés dans les résultats des instructions `EXPLAIN` dans Athena.

Types de sorties de l'instruction EXPLAIN

Les sorties de l'instruction `EXPLAIN` peuvent être de deux types :

- Logical plan (Plan logique) – Affiche le plan logique utilisé par le moteur SQL pour exécuter une instruction. La syntaxe correspondante est `EXPLAIN` ou `EXPLAIN (TYPE LOGICAL)`.
- Distributed plan (Plan distribué) – Affiche un plan d'exécution dans un environnement distribué. La sortie montre les fragments, qui sont des étapes de traitement. Chaque fragment de plan est traité par un ou plusieurs nœuds. Les données peuvent être échangées entre les nœuds qui traitent les fragments. La syntaxe correspondante est `EXPLAIN (TYPE DISTRIBUTED)`.

  Dans la sortie d'un plan distribué, les fragments (étapes de traitement) sont indiqués par `Fragment` *`number`* [*`fragment_type`*] , où *`number`* est un entier basé sur zéro et où *`fragment_type`* spécifie comment le fragment est exécuté par les nœuds. Les types de fragments, qui donnent des informations sur la disposition de l'échange de données (Data Exchange), sont décrits dans le tableau suivant.

  Types de fragments de plan distribué

Type de fragment	Description
`SINGLE`	Le fragment est exécuté sur un seul nœud.
`HASH`	Le fragment est exécuté sur un nombre fixe de nœuds. Les données d'entrée sont distribuées à l'aide d'une fonction de hachage.
`ROUND_ROBIN`	Le fragment est exécuté sur un nombre fixe de nœuds. Les données d'entrée sont distribuées de manière circulaire.

Type de fragment	Description
BROADCAST	Le fragment est exécuté sur un nombre fixe de nœuds. Les données d'entrée sont diffusées sur tous les nœuds.
SOURCE	Le fragment est exécuté sur les nœuds où les fractionnements d'entrée sont accessibles.

Exchange

Les termes liés à l'échange décrivent la façon dont les données sont échangées entre les composants master. Les transferts peuvent être locaux ou distants.

LocalExchange [*type_échange*]

Transfère les données localement dans les composants master pour les différentes étapes d'une requête. La valeur d'*exchange_type* peut être l'un des types d'échange logique ou distribué décrits plus loin dans cette section.

RemoteExchange [*type_échange*]

Transfert de données entre les composants master pour les différentes étapes d'une requête. La valeur d'*exchange_type* peut être l'un des types d'échange logique ou distribué décrits plus loin dans cette section.

Types d'échanges logiques

Les types d'échange suivants décrivent les actions prises pendant la phase d'échange d'un plan logique.

- **GATHER** – Un seul composant master rassemble la sortie de tous les autres composants master. Par exemple, la dernière étape d'une requête Select rassemble les résultats de tous les nœuds et écrit les résultats sur Simple Storage Service (Amazon S3).
- **REPARTITION** – Envoie les données de la ligne à un esclave spécifique en fonction du schéma de partitionnement requis pour s'appliquer à l'opérateur suivant.
- **REPLICATE** – Copie les données de ligne vers tous les esclaves.

Types d'échanges distribués

Les types d'échange suivants indiquent la disposition des données lorsqu'elles sont échangées entre les nœuds d'un plan distribué.

- **`HASH`** – L'échange distribue des données vers plusieurs destinations à l'aide d'une fonction de hachage.
- **`SINGLE`** – L'échange distribue les données vers une seule destination.

Analyse

Les conditions suivantes décrivent comment les données sont analysées au cours d'une requête.

TableScan

Analyse les données source d'une table à partir de Simple Storage Service (Amazon S3) ou d'un connecteur Apache Hive et applique l'élagage des partitions généré par le prédicat du filtre.

ScanFilter

Analyse les données source d'une table à partir de Simple Storage Service (Amazon S3) ou d'un connecteur Hive et applique l'élagage de partitions généré par le prédicat de filtre et par des prédicats de filtre supplémentaires non appliqués par l'élagage de partitions.

ScanFilterProject

Tout d'abord, il analyse les données source d'une table à partir de Simple Storage Service (Amazon S3) ou d'un connecteur Hive et applique l'élagage de partitions généré par le prédicat de filtre et par des prédicats de filtre supplémentaires non appliqués par l'élagage de partitions. Ensuite, il modifie la disposition de la mémoire des données de sortie en une nouvelle projection afin d'améliorer les performances des étapes ultérieures.

Joindre

Joint les données entre deux tables. Les jointures peuvent être classées par type de jointure et par type de distribution.

Types de jointures

Les types de jointure définissent la manière dont l'opération de jointure se produit.

CrossJoin— Produit le produit cartésien des deux tables jointes.

InnerJoin— Sélectionne les enregistrements dont les valeurs correspondent dans les deux tables.

LeftJoin— Sélectionne tous les enregistrements de la table de gauche et les enregistrements correspondants de la table de droite. Si aucune correspondance ne se produit, le résultat sur le côté droit est NULL.

RightJoin— Sélectionne tous les enregistrements de la table de droite et les enregistrements correspondants de la table de gauche. Si aucune correspondance ne se produit, le résultat sur le côté gauche est NULL.

FullJoin— Sélectionne tous les enregistrements présentant une correspondance dans les enregistrements de la table de gauche ou de droite. La table jointe contient tous les registres des tables et remplit les valeurs NULL pour les correspondances manquantes de chaque côté.

> **Note**
>
> Pour des raisons de performances, le moteur de requête peut réécrire une requête de jointure dans un type de jointure différent pour produire les mêmes résultats. Par exemple, une requête de jointure interne avec prédicat sur une table peut être réécrite dans un `CrossJoin`. Cela repousse le prédicat vers la phase de balayage de la table, de sorte que moins de données sont balayées.

Types de distributions des jointures

Les types de distribution définissent la façon dont les données sont échangées entre les composants master lorsque l'opération de jointure est exécutée.

Partitionné – Les tables de gauche et de droite sont partitionnées par hachage sur tous les composants master. La distribution partitionnée consomme moins de mémoire dans chaque nœud. La distribution partitionnée peut être beaucoup plus lente que les jointures répliquées. Les jointures partitionnées conviennent lorsque vous joignez deux grandes tables.

Répliqué – Une table est partitionnée par hachage sur tous les composants master et l'autre table est répliquée sur tous les composants master pour effectuer l'opération de jointure. La distribution répliquée peut être beaucoup plus rapide que les jointures partitionnées, mais elle consomme plus de mémoire dans chaque composant master. Si la table répliquée est trop grande, le nœud de travail peut rencontrer une out-of-memory erreur. Les jointures répliquées conviennent lorsque l'une des tables jointes est petite.

PREPARE

Crée une instruction SQL avec le nom `statement_name` à exécuter ultérieurement. L'instruction peut inclure des paramètres représentés par des points d'interrogation. Pour fournir des valeurs pour les paramètres et exécuter l'instruction préparée, utilisez EXECUTE.

Résumé

```
PREPARE statement_name FROM statement
```

La table suivante décrit ces paramètres.

Paramètre	Description
`statement_name`	Nom de l'instruction à préparer. Le nom doit être unique au sein du groupe de travail.
`statement`	Une requête `SELECT`, `CTAS` ou `INSERT INTO`.

> **Note**
>
> Le nombre maximal d'instructions préparées dans un groupe de travail est de 1 000.

Exemples

L'exemple suivant prépare une requête de sélection sans paramètre.

```
PREPARE my_select1 FROM
SELECT * FROM nation
```

L'exemple suivant prépare une requête de sélection qui inclut des paramètres. Les valeurs pour `productid` et `quantity` seront fournies par la clause `USING` d'une instruction `EXECUTE` :

```
PREPARE my_select2 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
```

L'exemple suivant prépare une requête d'insertion.

```
PREPARE my_insert FROM
INSERT INTO cities_usa (city, state)
SELECT city, state
FROM cities_world
WHERE country = ?
```

Ressources supplémentaires

Interrogation avec des instructions préparées

EXECUTE

DEALLOCATE PREPARE

INSERT INTO

EXECUTE

Exécute une instruction préparée portant le nom `statement_name`. Les valeurs des paramètres pour les points d'interrogation dans l'instruction préparée sont définies dans la clause `USING` dans une liste séparée par des virgules. Pour créer une instruction préparée, utilisez PREPARE.

Résumé

```
EXECUTE statement_name [ USING parameter1[, parameter2, ... ] ]
```

Exemples

L'exemple suivant prépare et exécute une requête sans paramètre.

```
PREPARE my_select1 FROM
SELECT name FROM nation
EXECUTE my_select1
```

L'exemple suivant prépare et exécute une requête avec un seul paramètre.

```
PREPARE my_select2 FROM
SELECT * FROM "my_database"."my_table" WHERE year = ?
EXECUTE my_select2 USING 2012
```

Cette expression est similaire à :

```
SELECT * FROM "my_database"."my_table" WHERE year = 2012
```

L'exemple suivant prépare et exécute une requête avec deux paramètres.

```
PREPARE my_select3 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
EXECUTE my_select3 USING 346078, 12
```

Ressources supplémentaires

Interrogation avec des instructions préparées

PREPARE

INSERT INTO

DEALLOCATE PREPARE

Supprime l'instruction préparée portant le nom spécifié des instructions préparées dans le groupe de travail actuel.

Résumé

```
DEALLOCATE PREPARE statement_name
```

Exemples

L'exemple suivant supprime l'instruction préparée `my_select1` du groupe de travail actuel.

```
DEALLOCATE PREPARE my_select1
```

Ressources supplémentaires

Interrogation avec des instructions préparées

PREPARE

UNLOAD

Écrit les résultats de requête à partir d'une instruction `SELECT` au format de données spécifié. Les formats pris en charge pour `UNLOAD` comprennent Apache Parquet, ORC, Apache Avro et JSON. Le format CSV est le seul format de sortie pris en charge par la `SELECT` commande Athena, mais

vous pouvez utiliser cette UNLOAD commande, qui prend en charge différents formats de sortie, pour inclure votre SELECT requête et réécrire sa sortie dans l'un des formats pris en charge. UNLOAD

Bien que vous puissiez utiliser l'instruction CTAS pour produire des données dans des formats autres que CSV, ces instructions nécessitent également la création d'une table dans Athena. L'instruction UNLOAD est utile lorsque vous voulez produire les résultats d'une requête SELECT dans un format non CSV, mais que vous n'avez pas besoin de la table associée. Par exemple, une application en aval peut exiger que les résultats d'une requête SELECT soient au format JSON, et Parquet ou ORC peut offrir un avantage de performance par rapport au CSV si vous avez l'intention d'utiliser les résultats de la requête SELECT pour une analyse supplémentaire..

Considérations et restrictions

Lorsque vous utilisez l'instruction UNLOAD dans Athena, gardez à l'esprit les points suivants :

- Aucun ordre global des fichiers – Les résultats UNLOAD sont écrits dans plusieurs fichiers en parallèle. Si la requête SELECT de l'instruction UNLOAD spécifie un ordre de tri, le contenu de chaque fichier est trié, mais les fichiers ne sont pas triés les uns par rapport aux autres.
- Données orphelines non supprimées – En cas d'échec, Athena ne tente pas de supprimer les données orphelines. Ce comportement est le même que pour les instructions CTAS et INSERT INTO.
- Partitions maximales – Le nombre maximal de partitions pouvant être utilisées avec UNLOAD est 100.
- Fichiers manifestes et métadonnées – Athena génère un fichier de métadonnées et un fichier manifeste de données pour chaque requête UNLOAD. Le manifeste suit les fichiers écrits par la requête. Les deux fichiers sont enregistrés dans votre emplacement de résultat de requête Athena dans Simple Storage Service (Amazon S3). Pour plus d'informations, consultez Identification des fichiers de sortie de requête.
- Chiffrement – Les fichiers de sortie UNLOAD sont chiffrés selon la configuration de chiffrement utilisée pour Simple Storage Service (Amazon S3). Pour configurer la configuration du chiffrement afin de chiffrer votre UNLOAD résultat, vous pouvez utiliser l'EncryptionConfiguration API.
- Instructions préparées – UNLOAD peut être utilisé avec des instructions préparées. Pour plus d'informations sur les instructions préparées dans Athena, voir Utilisation de requêtes paramétrées.
- Service Quotas – UNLOAD utilise des quotas de requête DML. Pour plus d'informations sur les quotas, voir Service Quotas.
- Propriétaire du compartiment attendu – Le paramètre propriétaire du compartiment attendu ne s'applique pas à l'emplacement de destination de Simple Storage Service (Amazon S3)

spécifié dans la requête UNLOAD. Le paramètre propriétaire du compartiment attendu s'applique uniquement à l'emplacement de sortie de Simple Storage Service (Amazon S3) que vous spécifiez pour les résultats de la requête Athena. Pour plus d'informations, consultez Spécification d'un emplacement de résultats de requête à l'aide de la console Athena.

Syntaxe

L'instruction UNLOAD utilise la syntaxe suivante.

```
UNLOAD (SELECT col_name[, ...] FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/my_folder/'
WITH ( property_name = 'expression' [, ...] )
```

Sauf lors de l'écriture sur des partitions, la TO destination doit spécifier un emplacement dans Amazon S3 qui ne contient aucune donnée. Avant que la requête UNLOAD n'écrive à l'emplacement spécifié, elle vérifie que l'emplacement du compartiment est vide. Comme UNLOAD n'écrit pas de données dans l'emplacement spécifié si celui-ci contient déjà des données, UNLOAD n'écrase pas les données existantes. Pour réutiliser un emplacement de compartiment comme destination pour UNLOAD, supprimez les données de l'emplacement de compartiment, puis exécutez à nouveau la requête.

Notez qu'en cas d'UNLOADécriture sur des partitions, ce comportement est différent. Si vous exécutez plusieurs fois la même UNLOAD requête avec la même SELECT instruction, le même TO emplacement et les mêmes partitions, chaque UNLOAD requête décharge les données dans Amazon S3 à l'emplacement et aux partitions spécifiés.

Paramètres

Les valeurs possibles pour *property_name* sont les suivantes.

format = '***file_format***'

Obligatoire. Spécifie le format de fichier de sortie. Les valeurs possibles pour *file_format* sont ORC, PARQUET, AVRO, JSON ou TEXTFILE.

compression = '***compression_format***'

Facultatif. Cette option est spécifique aux formats ORC et Parquet. Pour ORC, la valeur par défaut est zlib ; pour Parquet, la valeur par défaut est gzip. Pour plus d'informations sur les formats de compression pris en charge, consultez Prise en charge de la compression Athena.

> **Note**
>
> Cette option ne s'applique pas au format `AVRO`. Athena utilise `gzip` pour les formats `JSON` et `TEXTFILE`.

compression_level = ***`compression_level`***

Facultatif. Le niveau de compression à utiliser pour la compression ZSTD. Cette propriété s'applique uniquement à la compression ZSTD. Pour plus d'informations, consultez Utilisation des niveaux de compression ZSTD dans Athena.

field_delimiter = '***`delimiter`***'

Facultatif. Spécifie un délimiteur de champ à un seul caractère pour les fichiers au format CSV, TSV et autres formats texte. L'exemple suivant spécifie une virgule comme délimiteur.

```
WITH (field_delimiter = ',')
```

Actuellement, les délimiteurs de champ à plusieurs caractères ne sont pas pris en charge. Si vous ne spécifiez pas de délimiteur de champ, le caractère octal `\001` (^A) est utilisé.

partitioned_by = ARRAY[***`col_name`***[,…]]

Facultatif. Tableau composé de colonnes à l'aide duquel la sortie est partitionnée.

> **Note**
>
> Dans votre instruction `SELECT`, veillez à ce que les noms des colonnes partitionnées figurent en dernier dans votre liste de colonnes.

Exemples

L'exemple suivant écrit la sortie d'une requête `SELECT` à l'emplacement Simple Storage Service (Amazon S3) `s3://DOC-EXAMPLE-BUCKET/unload_test_1/` en utilisant le format JSON.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/unload_test_1/'
WITH (format = 'JSON')
```

L'exemple suivant écrit la sortie d'une requête SELECT au format Parquet en utilisant la compression Snappy.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET',compression = 'SNAPPY')
```

L'exemple suivant écrit quatre colonnes au format texte, la sortie étant divisée par la dernière colonne.

```
UNLOAD (SELECT name1, address1, comment1, key1 FROM table1)
TO 's3://DOC-EXAMPLE-BUCKET/ partitioned/'
WITH (format = 'TEXTFILE', partitioned_by = ARRAY['key1'])
```

L'exemple suivant décharge les résultats de la requête vers l'emplacement spécifié à l'aide du format de fichier Parquet, de la compression ZSTD et du niveau de compression ZSTD 4.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET', compression = 'ZSTD', compression_level = 4)
```

Ressources supplémentaires

- Simplifier vos pipelines ETL et ML à l'aide de la fonctionnalité Amazon Athena UNLOAD sur le blog AWS Big Data.

Fonctions dans Amazon Athena

Pour les modifications apportées aux fonctions entre les versions du moteur Athena, consultez le document Référence de la version du moteur Athena. Pour une liste des fuseaux horaires pouvant être utilisés avec l'opérateur AT TIME ZONE, voir Fuseaux horaires pris en charge.

Version 3 du moteur Athena

Les fonctions dans la version 3 du moteur Athena sont basées sur Trino. Pour plus d'informations sur les fonctions, les opérateurs et les expressions Trino, veuillez consulter la rubrique Functions and operators et les sous-sections suivantes de la documentation de Trino.

- Regrouper

- Tableau
- Binaire
- Bitwise
- Color (Couleur)
- Comparison (Comparaison)
- Conditionnel
- Conversion
- Date et heure
- Décimal
- Géospatial
- HyperLogLog
- Adresse IP
- JSON
- Lambda
- Logique
- Machine learning
- Map
- Math
- Digest quantile
- Expression régulière
- Session
- Set Digest
- String
- Tableau
- Teradata
- T-Digest
- URL
- UUID

- Fenêtre

fonction invoker_principal ()

Cette `invoker_principal` fonction est propre à la version 3 du moteur Athena et ne se trouve pas dans Trino.

Renvoie un `VARCHAR` qui contient l'ARN du principal (rôle IAM ou identité du centre d'identité) qui a exécuté la requête appelant la fonction. Par exemple, si l'invocateur de requête utilise les autorisations d'un rôle IAM pour exécuter la requête, la fonction renvoie l'ARN du rôle IAM. Le rôle qui exécute la requête doit autoriser l'`LakeFormation:GetDataLakePrincipal`action.

Utilisation

```
SELECT invoker_principal()
```

Le tableau suivant présente un exemple de résultat.

#	_col0
1	*arn:aws:iam : 111122223333 : rôle/administrateur*

Version 2 du moteur Athena

Les fonctions dans la version 2 du moteur Athena sont basées sur Presto 0.217. Pour les fonctions géospatiales de la version 2 du moteur Athena, voir Fonctions géospatiales de la version 2 du moteur Athena.

> **Note**
>
> La documentation spécifique à la version pour les fonctions de Presto 0.217 n'est plus disponible. Pour plus d'informations sur les fonctions, les opérateurs et les expressions Presto actuels, veuillez consulter la rubrique Fonctions et opérateurs Presto ou consultez les liens des sous-catégories dans cette section.

- Opérateurs logiques

- Fonctions et opérateurs de comparaison
- Expressions conditionnelles
- Fonctions de conversion
- Fonctions et opérateurs mathématiques
- Fonctions bitwise
- Fonctions et opérateurs décimaux
- Fonctions et opérateurs de chaînes de caractères
- Fonctions binaires
- Fonctions et opérateurs de la date et de l'heure
- Fonctions d'expression régulière
- Fonctions et opérateurs JSON
- Fonctions URL
- Fonctions d'agrégation
- Fonctions de fenêtrage
- Fonctions de couleur
- Opérateurs et fonctions de tableau
- Fonctions et opérateurs de mappage
- Expressions et fonctions lambda
- Fonctions Teradata

Fuseaux horaires pris en charge

Vous pouvez utiliser l'opérateur `AT TIME ZONE` dans une instruction `SELECT timestamp` pour spécifier le fuseau horaire pour l'horodatage qui est renvoyé, comme dans l'exemple suivant :

```
SELECT timestamp '2012-10-31 01:00 UTC' AT TIME ZONE 'America/Los_Angeles' AS la_time;
```

Résultats

```
la_time
```

```
2012-10-30 18:00:00.000 America/Los_Angeles
```

La liste suivante contient les fuseaux horaires qui peuvent être utilisés avec l'opérateur `AT TIME ZONE` dans Athena. Pour plus de fonctions et d'exemples concernant le fuseau horaire, consultez Fonctions de fuseau horaire et exemples.

```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Asmera
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Ceuta
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/El_Aaiun
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Juba
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
```

```
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Timbuktu
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Araguaina
America/Argentina/Buenos_Aires
America/Argentina/Catamarca
America/Argentina/ComodRivadavia
America/Argentina/Cordoba
America/Argentina/Jujuy
America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/Salta
America/Argentina/San_Juan
America/Argentina/San_Luis
America/Argentina/Tucuman
America/Argentina/Ushuaia
America/Aruba
America/Asuncion
America/Atikokan
America/Atka
America/Bahia
America/Bahia_Banderas
America/Barbados
America/Belem
America/Belize
```

```
America/Blanc-Sablon
America/Boa_Vista
America/Bogota
America/Boise
America/Buenos_Aires
America/Cambridge_Bay
America/Campo_Grande
America/Cancun
America/Caracas
America/Catamarca
America/Cayenne
America/Cayman
America/Chicago
America/Chihuahua
America/Coral_Harbour
America/Cordoba
America/Costa_Rica
America/Creston
America/Cuiaba
America/Curacao
America/Danmarkshavn
America/Dawson
America/Dawson_Creek
America/Denver
America/Detroit
America/Dominica
America/Edmonton
America/Eirunepe
America/El_Salvador
America/Ensenada
America/Fort_Nelson
America/Fort_Wayne
America/Fortaleza
America/Glace_Bay
America/Godthab
America/Goose_Bay
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
```

```
America/Hermosillo
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Petersburg
America/Indiana/Tell_City
America/Indiana/Vevay
America/Indiana/Vincennes
America/Indiana/Winamac
America/Indianapolis
America/Inuvik
America/Iqaluit
America/Jamaica
America/Jujuy
America/Juneau
America/Kentucky/Louisville
America/Kentucky/Monticello
America/Knox_IN
America/Kralendijk
America/La_Paz
America/Lima
America/Los_Angeles
America/Louisville
America/Lower_Princes
America/Maceio
America/Managua
America/Manaus
America/Marigot
America/Martinique
America/Matamoros
America/Mazatlan
America/Mendoza
America/Menominee
America/Merida
America/Metlakatla
America/Mexico_City
America/Miquelon
America/Moncton
America/Monterrey
America/Montevideo
America/Montreal
America/Montserrat
America/Nassau
America/New_York
```

```
America/Nipigon
America/Nome
America/Noronha
America/North_Dakota/Beulah
America/North_Dakota/Center
America/North_Dakota/New_Salem
America/Ojinaga
America/Panama
America/Pangnirtung
America/Paramaribo
America/Phoenix
America/Port-au-Prince
America/Port_of_Spain
America/Porto_Acre
America/Porto_Velho
America/Puerto_Rico
America/Punta_Arenas
America/Rainy_River
America/Rankin_Inlet
America/Recife
America/Regina
America/Resolute
America/Rio_Branco
America/Rosario
America/Santa_Isabel
America/Santarem
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/Shiprock
America/Sitka
America/St_Barthelemy
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Swift_Current
America/Tegucigalpa
America/Thule
America/Thunder_Bay
America/Tijuana
America/Toronto
```

```
America/Tortola
America/Vancouver
America/Virgin
America/Whitehorse
America/Winnipeg
America/Yakutat
America/Yellowknife
Antarctica/Casey
Antarctica/Davis
Antarctica/DumontDUrville
Antarctica/Macquarie
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
Antarctica/Rothera
Antarctica/South_Pole
Antarctica/Syowa
Antarctica/Troll
Antarctica/Vostok
Arctic/Longyearbyen
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Atyrau
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Barnaul
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Calcutta
Asia/Chita
Asia/Choibalsan
Asia/Chongqing
Asia/Chungking
Asia/Colombo
Asia/Dacca
```

```
Asia/Damascus
Asia/Dhaka
Asia/Dili
Asia/Dubai
Asia/Dushanbe
Asia/Gaza
Asia/Harbin
Asia/Hebron
Asia/Ho_Chi_Minh
Asia/Hong_Kong
Asia/Hovd
Asia/Irkutsk
Asia/Istanbul
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Kashgar
Asia/Kathmandu
Asia/Katmandu
Asia/Khandyga
Asia/Kolkata
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuching
Asia/Kuwait
Asia/Macao
Asia/Macau
Asia/Magadan
Asia/Makassar
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novokuznetsk
Asia/Novosibirsk
Asia/Omsk
Asia/Oral
Asia/Phnom_Penh
Asia/Pontianak
Asia/Pyongyang
Asia/Qatar
Asia/Qyzylorda
```

```
Asia/Rangoon
Asia/Riyadh
Asia/Saigon
Asia/Sakhalin
Asia/Samarkand
Asia/Seoul
Asia/Shanghai
Asia/Singapore
Asia/Srednekolymsk
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Tel_Aviv
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Tomsk
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Urumqi
Asia/Ust-Nera
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yangon
Asia/Yekaterinburg
Asia/Yerevan
Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Faroe
Atlantic/Jan_Mayen
Atlantic/Madeira
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
Australia/ACT
Australia/Adelaide
Australia/Brisbane
```

```
Australia/Broken_Hill
Australia/Canberra
Australia/Currie
Australia/Darwin
Australia/Eucla
Australia/Hobart
Australia/LHI
Australia/Lindeman
Australia/Lord_Howe
Australia/Melbourne
Australia/NSW
Australia/North
Australia/Perth
Australia/Queensland
Australia/South
Australia/Sydney
Australia/Tasmania
Australia/Victoria
Australia/West
Australia/Yancowinna
Brazil/Acre
Brazil/DeNoronha
Brazil/East
Brazil/West
CET
CST6CDT
Canada/Atlantic
Canada/Central
Canada/Eastern
Canada/Mountain
Canada/Newfoundland
Canada/Pacific
Canada/Saskatchewan
Canada/Yukon
Chile/Continental
Chile/EasterIsland
Cuba
EET
EST5EDT
Egypt
Eire
Europe/Amsterdam
Europe/Andorra
Europe/Astrakhan
```

```
Europe/Athens
Europe/Belfast
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Busingen
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Guernsey
Europe/Helsinki
Europe/Isle_of_Man
Europe/Istanbul
Europe/Jersey
Europe/Kaliningrad
Europe/Kiev
Europe/Kirov
Europe/Lisbon
Europe/Ljubljana
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Mariehamn
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Nicosia
Europe/Oslo
Europe/Paris
Europe/Podgorica
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/San_Marino
Europe/Sarajevo
Europe/Simferopol
Europe/Skopje
Europe/Sofia
```

```
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Tiraspol
Europe/Ulyanovsk
Europe/Uzhgorod
Europe/Vaduz
Europe/Vatican
Europe/Vienna
Europe/Vilnius
Europe/Volgograd
Europe/Warsaw
Europe/Zagreb
Europe/Zaporozhye
Europe/Zurich
GB
GB-Eire
Hongkong
Iceland
Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
Iran
Israel
Jamaica
Japan
Kwajalein
Libya
MET
MST7MDT
Mexico/BajaNorte
Mexico/BajaSur
Mexico/General
NZ
NZ-CHAT
Navajo
```

```
PRC
PST8PDT
Pacific/Apia
Pacific/Auckland
Pacific/Bougainville
Pacific/Chatham
Pacific/Chuuk
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofo
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Johnston
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Kwajalein
Pacific/Majuro
Pacific/Marquesas
Pacific/Midway
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau
Pacific/Pitcairn
Pacific/Pohnpei
Pacific/Ponape
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
Pacific/Samoa
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk
Pacific/Wake
Pacific/Wallis
```

```
Pacific/Yap
Poland
Portugal
ROK
Singapore
Turkey
US/Alaska
US/Aleutian
US/Arizona
US/Central
US/East-Indiana
US/Eastern
US/Hawaii
US/Indiana-Starke
US/Michigan
US/Mountain
US/Pacific
US/Pacific-New
US/Samoa
W-SU
WET
```

Fonctions de fuseau horaire et exemples

Vous trouverez ci-dessous d'autres fonctions et exemples liés aux fuseaux horaires.

- at_timezone (***horodatage***, ***zone***) – Renvoie la valeur de *horodatage* dans l'heure locale correspondante pour *zone*.

  Exemple

  ```
  SELECT at_timezone(timestamp '2021-08-22 00:00 UTC', 'Canada/Newfoundland')
  ```

  Result

  ```
  2021-08-21 21:30:00.000 Canada/Newfoundland
  ```

- timezone_hour (***horodatage***) – Renvoie l'heure du décalage horaire par rapport à l'horodatage sous forme de `bigint`.

  Exemple

```
SELECT timezone_hour(timestamp '2021-08-22 04:00 UTC' AT TIME ZONE 'Canada/
Newfoundland')
```

Result

```
-2
```

- timezone_minute(***horodatage***) – Renvoie la minute du décalage horaire par rapport à l'*horodatage* sous la forme d'un `bigint`.

  Exemple

  ```
  SELECT timezone_minute(timestamp '2021-08-22 04:00 UTC' AT TIME ZONE 'Canada/
  Newfoundland')
  ```

  Result

  ```
  -30
  ```

- with_timezone(***horodatage***, ***zone***) – Renvoie un timestamp avec un fuseau horaire à partir des valeurs de l'*horodatage* et de la *zone* spécifiées.

  Exemple

  ```
  SELECT with_timezone(timestamp '2021-08-22 04:00', 'Canada/Newfoundland')
  ```

  Result

  ```
  2021-08-22 04:00:00.000 Canada/Newfoundland
  ```

Instructions DDL

Utilisez les instructions DDL suivantes directement dans Athena.

Le moteur de requête Athena est basé partiellement sur HiveQL DDL.

Athena ne prend pas en charge toutes les instructions DDL et il existe quelques différences entre le DDL HiveQL et le DDL Athena. Pour plus d'informations, consultez les rubriques de référence de cette section et DDL non prise en charge.

Rubriques

- SHOW VIEWS

DDL non prise en charge

Les instructions DDL suivantes ne sont pas prises en charge par Athena :

- ALTER INDEX
- ALTER TABLE *table_name* ARCHIVE PARTITION
- ALTER TABLE *table_name* CLUSTERED BY
- ALTER TABLE *table_name* EXCHANGE PARTITION
- ALTER TABLE *table_name* NOT CLUSTERED
- ALTER TABLE *table_name* NOT SKEWED
- ALTER TABLE *table_name* NOT SORTED
- ALTER TABLE *table_name* NOT STORED AS DIRECTORIES
- ALTER TABLE *table_name* partitionSpec CHANGE COLUMNS
- ALTER TABLE *table_name* partitionSpec COMPACT
- ALTER TABLE *table_name* partitionSpec CONCATENATE
- ALTER TABLE *table_name* partitionSpec SET FILEFORMAT
- ALTER TABLE *table_name* SET SERDEPROPERTIES
- ALTER TABLE *table_name* SET SKEWED LOCATION
- ALTER TABLE *table_name* SKEWED BY
- ALTER TABLE *table_name* TOUCH
- ALTER TABLE *table_name* UNARCHIVE PARTITION
- COMMIT
- CREATE INDEX
- CREATE ROLE
- CREATE TABLE *table_name* LIKE *existing_table_name*
- CREATE TEMPORARY MACRO
- DELETE FROM
- DESCRIBE DATABASE
- DFS

- DROP INDEX
- DROP ROLE
- DROP TEMPORARY MACRO
- EXPORT TABLE
- GRANT ROLE
- IMPORT TABLE
- LOCK DATABASE
- LOCK TABLE
- REVOKE ROLE
- ROLLBACK
- SHOW COMPACTIONS
- SHOW CURRENT ROLES
- SHOW GRANT
- SHOW INDEXES
- SHOW LOCKS
- SHOW PRINCIPALS
- SHOW ROLE GRANT
- SHOW ROLES
- SHOW STATS
- SHOW TRANSACTIONS
- START TRANSACTION
- UNLOCK DATABASE
- UNLOCK TABLE

ALTER DATABASE SET DBPROPERTIES

Crée une ou plusieurs propriétés pour une base de données. `DATABASE` et `SCHEMA` sont interchangeables ; ils signifient la même chose.

Résumé

```
ALTER {DATABASE|SCHEMA} database_name
```

```
  SET DBPROPERTIES ('property_name'='property_value' [, ...] )
```

Paramètres

SET DBPROPERTIES ('property_name'='property_value' [, ...]

Spécifie une ou des propriétés pour la base de données nommée `property_name` et établit la valeur de chacune des propriétés respectivement comme `property_value`. Si `property_name` existe déjà, l'ancienne valeur est remplacée par `property_value`.

Exemples

```
ALTER DATABASE jd_datasets
  SET DBPROPERTIES ('creator'='John Doe', 'department'='applied mathematics');
```

```
ALTER SCHEMA jd_datasets
  SET DBPROPERTIES ('creator'='Jane Doe');
```

ALTER TABLE ADD COLUMNS

Ajoute une ou plusieurs colonnes à une table existante. Lorsque la syntaxe facultative `PARTITION` est utilisée, met à jour les métadonnées de partition.

Résumé

```
ALTER TABLE table_name
  [PARTITION
   (partition_col1_name = partition_col1_value
   [,partition_col2_name = partition_col2_value][,...])]
  ADD COLUMNS (col_name data_type)
```

Paramètres

PARTITION (partition_col_name = partition_col_value [,...])

Crée une partition avec les combinaisons nom/valeur de colonne que vous spécifiez. Placez `partition_col_value` entre guillemets uniquement si le type de données de la colonne est une chaîne.

AJOUTER DES COLONNES (col_name data_type [,col_name data_type,...])

Ajoute des colonnes après les colonnes existantes, mais avant les colonnes de partition.

Exemples

```
ALTER TABLE events ADD COLUMNS (eventowner string)
```

```
ALTER TABLE events PARTITION (awsregion='us-west-2') ADD COLUMNS (event string)
```

```
ALTER TABLE events PARTITION (awsregion='us-west-2') ADD COLUMNS (eventdescription
 string)
```

Remarques

- Pour afficher une nouvelle colonne de table dans le panneau de navigation de l'éditeur de requête Athena après l'exécution de `ALTER TABLE ADD COLUMNS`, actualisez manuellement la liste des tables dans l'éditeur, puis développez à nouveau la table.
- `ALTER TABLE ADD COLUMNS` ne fonctionne pas pour les colonnes avec le type de données `date`. Pour contourner ce problème, utilisez le type de données `timestamp` à la place.

ALTER TABLE ADD PARTITION

Crée une ou plusieurs colonnes de partition pour la table. Chaque partition se compose d'une ou plusieurs combinaisons nom/valeur de colonne distinctes. Un répertoire de données distinct est créé pour chaque combinaison spécifiée, ce qui peut améliorer les performances des requêtes dans certaines circonstances. Les colonnes partitionnées n'existent pas dans les données de table elles-mêmes. Par conséquent, si vous utilisez un nom de colonne qui porte le même nom qu'une colonne dans la table elle-même, vous obtenez une erreur. Pour plus d’informations, consultez Partitionnement de données dans Athena.

Dans Athena, une table et ses partitions doivent utiliser les mêmes formats de données, mais leurs schémas peuvent différer. Pour plus d’informations, consultez Mises à jour dans les tables avec des partitions.

Pour plus d'informations sur les autorisations au niveau des ressources requises dans les politiques IAM (notamment `glue:CreatePartition`), consultez les rubriques Autorisations d'API

AWS Glue : référence des actions et ressources et Accès détaillé aux bases de données et aux tables du AWS Glue Data Catalog. Pour des informations de résolution de problèmes concernant les autorisations lors de l'utilisation d'Athena, consultez la section Autorisations de la rubrique Résolution des problèmes dans Athena.

Résumé

```
ALTER TABLE table_name ADD [IF NOT EXISTS]
  PARTITION
  (partition_col1_name = partition_col1_value
  [,partition_col2_name = partition_col2_value]
  [,...])
  [LOCATION 'location1']
  [PARTITION
  (partition_colA_name = partition_colA_value
  [,partition_colB_name = partition_colB_value
  [,...])]
  [LOCATION 'location2']
  [,...]
```

Paramètres

Lorsque vous ajoutez une partition, vous spécifiez une ou plusieurs paires nom/valeur de colonnes pour la partition et le chemin d'accès Simple Storage Service (Amazon S3) où résident les fichiers de données de cette partition.

[IF NOT EXISTS]

Entraîne la suppression de l'erreur si une partition avec la même définition existe déjà.

PARTITION (partition_col_name = partition_col_value [,...])

Crée une partition avec les combinaisons nom/valeur de colonne que vous spécifiez. Placez `partition_col_value` dans des caractères de chaîne uniquement si le type de données de la colonne est une chaîne.

[LOCATION 'emplacement']

Spécifie le répertoire dans lequel la partition définie par l'instruction précédente doit être stockée. La clause `LOCATION` est facultative lorsque les données utilisent le partitionnement de style Hive (`pk1=v1/pk2=v2/pk3=v3`). Grâce au partitionnement de style Hive, l'URI Amazon S3 complet est créé automatiquement à partir de l'emplacement de la table, des noms des clés de partition et

des valeurs des clés de partition. Pour plus d'informations, consultez Partitionnement de données dans Athena.

Considérations

Amazon Athena n'impose pas de limite spécifique au nombre de partitions que vous pouvez ajouter dans une seule instruction `ALTER TABLE ADD PARTITION` DDL. Toutefois, si vous devez ajouter un nombre important de partitions, pensez à diviser l'opération en lots plus petits afin d'éviter d'éventuels problèmes de performances. L'exemple suivant utilise des commandes successives pour ajouter des partitions individuellement et `IF NOT EXISTS` pour éviter d'ajouter des doublons.

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITION (ds='2023-01-01')
ALTER TABLE table_name ADD IF NOT EXISTS PARTITION (ds='2023-01-02')
ALTER TABLE table_name ADD IF NOT EXISTS PARTITION (ds='2023-01-03')
```

Lorsque vous travaillez avec des partitions dans Athena, gardez également à l'esprit les points suivants :

- Bien qu'Athena prenne en charge l'interrogation de AWS Glue tables contenant 10 millions de partitions, Athena ne peut pas lire plus d'un million de partitions en un seul scan.
- Pour optimiser vos requêtes et réduire le nombre de partitions scannées, envisagez des stratégies telles que l'élagage des partitions ou l'utilisation d'index de partition.
- Si vous n'en utilisez pas AWS Glue Data Catalog, le nombre maximum de partitions par table est de 20 000. Vous pouvez demander une augmentation de quota.

Pour des considérations supplémentaires concernant l'utilisation des partitions dans Athena, voir. Partitionnement de données dans Athena

Exemples

L'exemple suivant ajoute une partition unique à une table pour les données partitionnées de style Hive.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-14', country = 'IN');
```

L'exemple suivant ajoute plusieurs partitions à une table pour les données partitionnées de style Hive.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-31', country = 'IN')
  PARTITION (dt = '2016-06-01', country = 'IN');
```

Lorsque la table n'est pas destinée à des données partitionnées de style Hive, la clause `LOCATION` est obligatoire et doit être l'URI Amazon S3 complet pour le préfixe contenant les données de la partition.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-31', country = 'IN') LOCATION 's3://DOC-EXAMPLE-BUCKET/path/
to/INDIA_31_May_2016/'
  PARTITION (dt = '2016-06-01', country = 'IN') LOCATION 's3://DOC-EXAMPLE-BUCKET/path/
to/INDIA_01_June_2016/';
```

Pour ignorer les erreurs lorsque la partition existe déjà, utilisez la clause `IF NOT EXISTS` comme dans l'exemple suivant.

```
ALTER TABLE orders ADD IF NOT EXISTS
  PARTITION (dt = '2016-05-14', country = 'IN');
```

Fichiers **`_$folder$`** zéro octet

Si vous exécutez une instruction `ALTER TABLE ADD PARTITION` et spécifiez par erreur une partition déjà existante et un emplacement Simple Storage Service (Amazon S3) incorrect, des fichiers d'emplacement zéro octet du format *`partition_value`*`_$folder$` sont créés dans Simple Storage Service (Amazon S3). Vous devez supprimer ces fichiers manuellement.

Pour éviter que cela ne se produise, utilisez la syntaxe `ADD IF NOT EXISTS` de votre instruction `ALTER TABLE ADD PARTITION`, comme dans l'exemple suivant.

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITION […]
```

ALTER TABLE DROP PARTITION

Supprime une ou plusieurs partitions spécifiées pour la table nommée.

Résumé

```
ALTER TABLE table_name DROP [IF EXISTS] PARTITION (partition_spec) [, PARTITION
 (partition_spec)]
```

Paramètres

[IF EXISTS]

Supprime le message d'erreur si la partition spécifiée n'existe pas.

PARTITION (partition_spec)

Chaque `partition_spec` indique une combinaison nom/valeur de colonne au format `partition_col_name = partition_col_value [,...]`.

Exemples

```
ALTER TABLE orders
DROP PARTITION (dt = '2014-05-14', country = 'IN');
```

```
ALTER TABLE orders
DROP PARTITION (dt = '2014-05-14', country = 'IN'), PARTITION (dt = '2014-05-15',
 country = 'IN');
```

Remarques

L'instruction `ALTER TABLE DROP PARTITION` ne fournit pas de syntaxe unique pour supprimer toutes les partitions en une seule fois ni ne prend en charge les critères de filtrage pour spécifier une gamme de partitions à supprimer.

Pour contourner le problème, vous pouvez utiliser l' AWS Glue API GetPartitionset les BatchDeletePartitionactions dans les scripts. L'action `GetPartitions` prend en charge des expressions de filtre complexes comme celles d'une expression SQL `WHERE`. Après avoir utilisé `GetPartitions` pour créer une liste filtrée de partitions à supprimer, vous pouvez utiliser l'action `BatchDeletePartition` pour supprimer les partitions par lots de 25.

> **Important**
>
> En raison d'un problème connu, lorsqu'une partition non valide est spécifiée pour l'instruction `ALTER TABLE DROP PARTITION`, toutes les partitions de la table sont supprimées dans AWS Glue. Par exemple, l'instruction suivante supprimera toutes les partitions de la table *`my_table`* même si la partition spécifiée n'existe pas. Pour contourner le problème, assurez-vous de saisir correctement les informations de partition avant d'exécuter l'instruction `ALTER TABLE DROP PARTITION`.

```
ALTER TABLE my_table DROP IF EXISTS PARTITION(zzz='');
```

ALTER TABLE RENAME PARTITION

Renomme la valeur d'une partition.

> **Note**
>
> ALTER TABLE RENAME PARTITIONne renomme pas les colonnes de partition. Pour modifier le nom d'une colonne de partition, vous pouvez utiliser la AWS Glue console. Pour plus d'informations, consultez Renommer une colonne de partition dans AWS Glue dans la suite de ce document.

Résumé

Pour la table nomméetable_name, renomme la valeur de partition spécifiée par `partition_spec` la valeur spécifiée parnew_partition_spec.

```
ALTER TABLE table_name PARTITION (partition_spec) RENAME TO PARTITION
 (new_partition_spec)
```

Paramètres

PARTITION (partition_spec)

Chaque `partition_spec` indique une combinaison nom/valeur de colonne au format `partition_col_name = partition_col_value [,...]`.

Exemples

```
ALTER TABLE orders
PARTITION (dt = '2014-05-14', country = 'IN') RENAME TO PARTITION (dt = '2014-05-15',
 country = 'IN');
```

Renommer une colonne de partition dans AWS Glue

Utilisez la procédure suivante pour renommer les noms des colonnes de partition dans la AWS Glue console.

Pour renommer une colonne de partition de table dans la console AWS Glue

1. Connectez-vous à la AWS Glue console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/glue/.
2. Dans le volet de navigation, choisissez Tables.
3. Sur la page Tables, utilisez la zone de recherche Filtrer les tables pour trouver la table que vous souhaitez modifier.
4. Dans la colonne Nom, choisissez le lien de la table que vous souhaitez modifier.
5. Sur la page de détails de la table, dans la section Schéma, effectuez l'une des opérations suivantes :
   - Pour modifier le nom au format JSON, choisissez Modifier le schéma au format JSON.
   - Pour modifier le nom directement, choisissez Modifier le schéma. Cette procédure sélectionne Modifier le schéma.
6. Cochez la case correspondant à la colonne partitionnée que vous souhaitez renommer, puis choisissez Modifier.
7. Dans la boîte de dialogue Modifier l'entrée du schéma, dans Nom, entrez le nouveau nom de la colonne de partition.
8. Choisissez Enregistrer en tant que nouvelle version du tableau. Cette action met à jour le nom de la colonne de partition et préserve l'historique de l'évolution du schéma sans créer de copie physique séparée de vos données.
9. Pour comparer les versions d'un tableau, sur la page de détails du tableau, sélectionnez Actions, puis Comparez les versions.

Ressources supplémentaires

Pour plus d'informations sur le partitionnement, consultez Partitionnement de données dans Athena.

ALTER TABLE REPLACE COLUMNS

Supprime toutes les colonnes existantes d'une table créée avec le LazySimpleSerDeet les remplace par le jeu de colonnes spécifié. Lorsque la syntaxe facultative `PARTITION` est utilisée, met à jour les

métadonnées de partition. Vous pouvez également utiliser `ALTER TABLE REPLACE COLUMNS` pour supprimer des colonnes en spécifiant uniquement les colonnes que vous souhaitez conserver.

Résumé

```
ALTER TABLE table_name
  [PARTITION
   (partition_col1_name = partition_col1_value
   [,partition_col2_name = partition_col2_value][,...])]
  REPLACE COLUMNS (col_name data_type [, col_name data_type, ...])
```

Paramètres

PARTITION (partition_col_name = partition_col_value [,...])

Spécifie une partition avec les combinaisons nom de colonne/valeur que vous spécifiez. Placez `partition_col_value` entre guillemets uniquement si le type de données de la colonne est une chaîne.

REMPLACER LES COLONNES (col_name data_type [,col_name data_type,...])

Remplace les colonnes existantes par les noms de colonnes et les types de données spécifiés.

Remarques

- Pour consulter la modification des colonnes de la table dans le panneau de navigation de l'éditeur de requêtes Athena après avoir exécuté `ALTER TABLE REPLACE COLUMNS`, vous devrez peut-être actualiser manuellement la liste des tables dans l'éditeur, puis développer à nouveau la table.
- `ALTER TABLE REPLACE COLUMNS` ne fonctionne pas pour les colonnes avec le type de données `date`. Pour contourner ce problème, utilisez le type de données `timestamp` dans la table à la place.
- Notez que même si vous ne remplacez qu'une seule colonne, la syntaxe doit être `ALTER TABLE` *`table-name`* `REPLACE COLUMNS`, avec COLUMNS au pluriel. Vous devez spécifier non seulement la colonne que vous souhaitez remplacer, mais aussi les colonnes que vous souhaitez conserver. Sinon, les colonnes que vous ne spécifiez pas seront supprimées. Cette syntaxe et ce comportement proviennent du DDL Apache Hive. Pour référence, consultez la section Add/Replace columns -Ajouter/remplacer des colonnes) dans la documentation Apache.

Exemple

Dans l'exemple suivant, la tablenames_cities, créée à l'aide du LazySimpleSerDe, comporte trois colonnes nommées col1col2, etcol3. Toutes les colonnes sont de type string. Pour afficher les colonnes du tableau, la commande suivante utilise la commande SHOW COLUMNS suivante.

```
SHOW COLUMNS IN names_cities
```

Résultat de la requête :

```
col1
col2
col3
```

La commande ALTER TABLE REPLACE COLUMNS suivante remplace les noms des colonnes par first_name, last_name et city. Les données source sous-jacentes ne sont pas affectées.

```
ALTER TABLE names_cities
REPLACE COLUMNS (first_name string, last_name string, city string)
```

Pour tester le résultat, SHOW COLUMNS est exécuté à nouveau.

```
SHOW COLUMNS IN names_cities
```

Résultat de la requête :

```
first_name
last_name
city
```

Une autre façon d'afficher les nouveaux noms de colonnes consiste à prévisualiser la table dans l'éditeur de requêtes Athena ou d'exécuter votre propre requête SELECT.

ALTER TABLE SET LOCATION

Modifie l'emplacement de la table nommée table_name et, éventuellement, une partition avec partition_spec.

Résumé

```
ALTER TABLE table_name [ PARTITION (partition_spec) ] SET LOCATION 'new location'
```

Paramètres

PARTITION (partition_spec)

Spécifie la partition avec les paramètres `partition_spec` dont vous voulez modifier l'emplacement. La spécification `partition_spec` définit une combinaison nom/valeur de colonne sous la forme `partition_col_name = partition_col_value`.

SET LOCATION 'nouvel emplacement'

Spécifie le nouvel emplacement, qui doit être un emplacement Simple Storage Service (Amazon S3). Pour plus d'informations sur la syntaxe, consultez la rubrique Emplacement de table dans Simple Storage Service (Amazon S3).

Exemples

```
ALTER TABLE customers PARTITION (zip='98040', state='WA') SET LOCATION 's3://DOC-
EXAMPLE-BUCKET/custdata/';
```

ALTER TABLE SET TBLPROPERTIES

Ajoute des propriétés de métadonnées personnalisées ou prédéfinies à une table et définit les valeurs qui leur sont attribuées. Pour afficher les propriétés d'une table, utilisez la commande SHOW TBLPROPERTIES.

Les Tables gérées par Apache Hive ne sont pas prises en charge, le paramètre `'EXTERNAL'='FALSE'` n'a donc aucun effet.

Résumé

```
ALTER TABLE table_name SET TBLPROPERTIES ('property_name' = 'property_value' [ , ... ])
```

Paramètres

SET TBLPROPERTIES ('nom_propriété' = 'valeur_propriété' [, ...])

Spécifie les propriétés de métadonnées à ajouter en tant que `property_name` et la valeur pour chacune d'entre elles en tant que `property value`. Si `property_name` existe déjà, sa valeur est définie sur la valeur `property_value` nouvellement spécifiée.

Les propriétés de table prédéfinies suivantes ont des utilisations spéciales.

Propriété prédéfinie	Description
`classification`	Indique le type de données pour AWS Glue. Les valeurs possibles sont `csv`, `parquet`, `orc`, `avro` ou `json`. Les tables créées pour Athena dans la CloudTrail console sont ajoutées en `cloudtrail` tant que valeur à la `classification` propriété. Pour plus d'informations, voir la section TBLPROPERTIES de CREATE TABLE.
`has_encrypted_data`	Indique si le jeu de données spécifié par `LOCATION` est chiffré. Pour plus d'informations, voir la section TBLPROPERTIES de CREATE TABLE et Création de tables basées sur des ensembles de données chiffrés dans Simple Storage Service (Amazon S3).
`orc.compress`	Spécifie un format de compression pour les données au format ORC. Pour plus d'informations, consultez ORC SerDe.
`parquet.compression`	Spécifie un format de compression pour les données au format Parquet. Pour plus d'informations, consultez Parquet SerDe.
`write.compression`	Spécifie un format de compression pour les données au format JSON ou fichier texte. Pour les formats Parquet et ORC, utilisez les propriétés `parquet.compression` et `orc.compress` , respectivement.
`compression_level`	Spécifie le niveau de compression à utiliser. Cette propriété s'applique uniquement à la compression ZSTD. Les valeurs possibles sont comprises entre 1 et 22. La valeur par défaut est 3. Pour plus d'informations, consultez Utilisation des niveaux de compression ZSTD dans Athena.

Propriété prédéfinie	Description
`projection.*`	Propriétés personnalisées utilisées dans la projection de partition qui permettent à Athena de savoir à quels modèles de partition s'attendre lorsqu'il exécute une requête sur une table. Pour plus d'informations, consultez Projection de partition avec Amazon Athena.
`skip.header.line.count`	Ignore les en-têtes dans les données lorsque vous définissez une table. Pour plus d'informations, consultez Omission des en-têtes.
`storage.location.template`	Spécifie un modèle de chemin Simple Storage Service (Amazon S3) personnalisé pour les partitions projetées. Pour plus d'informations, consultez Configuration de la projection de partition.

Exemples

L'exemple suivant ajoute une note de commentaire aux propriétés de la table.

```
ALTER TABLE orders
SET TBLPROPERTIES ('notes'="Please don't drop this table.");
```

L'exemple suivant modifie la table `existing_table` pour utiliser le format de fichier Parquet avec une compression ZSTD et un niveau de compression ZSTD 4.

```
ALTER TABLE existing_table
SET TBLPROPERTIES ('parquet.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE DATABASE

Crée une base de données. Les éléments `DATABASE` et `SCHEMA` sont interchangeables. Ils ont la même signification.

> **Note**
>
> En guise d'exemple en matière de création d'une base de données, de création d'une table et d'exécution d'une requête `SELECT` sur la table dans Athena, voir Mise en route.

Résumé

```
CREATE {DATABASE|SCHEMA} [IF NOT EXISTS] database_name
  [COMMENT 'database_comment']
  [LOCATION 'S3_loc']
  [WITH DBPROPERTIES ('property_name' = 'property_value') [, ...]]
```

Paramètres

[IF NOT EXISTS]

Entraîne la suppression de l'erreur s'il existe déjà une base de données nommée `database_name`.

[COMMENT database_comment]

Définit la valeur de métadonnées pour la propriété de métadonnées intégrée nommée `comment` et la valeur que vous fournissez pour `database_comment`. Dans AWS Glue, le `COMMENT` contenu est écrit dans le `Description` champ des propriétés de la base de données.

[LOCATION S3_loc]

Spécifie que l'emplacement des fichiers de base de données et du metastore est exprimé sous la forme `S3_loc`. Il doit s'agir d'un emplacement Simple Storage Service (Amazon S3).

[WITH DBPROPERTIES ('property_name' = 'property_value') [, ...]]

Permet de spécifier des propriétés de métadonnées personnalisées pour la définition de base de données.

Exemples

```
CREATE DATABASE clickstreams;
```

```
CREATE DATABASE IF NOT EXISTS clickstreams
  COMMENT 'Site Foo clickstream data aggregates'
  LOCATION 's3://DOC-EXAMPLE-BUCKET/clickstreams/'
  WITH DBPROPERTIES ('creator'='Jane D.', 'Dept.'='Marketing analytics');
```

Affichage des propriétés d'une base de données

Pour afficher les propriétés d'une base de données que vous créez en AWSDataCatalog utilisantCREATE DATABASE, vous pouvez utiliser la AWS CLI commande `aws glue get-database`, comme dans l'exemple suivant :

```
aws glue get-database --name <your-database-name>
```

Le code JSON de sortie se présente comme suit :

```
{
    "Database": {
        "Name": "<your-database-name>",
        "Description": "<your-database-comment>",
        "LocationUri": "s3://DOC-EXAMPLE-BUCKET",
        "Parameters": {
            "<your-database-property-name>": "<your-database-property-value>"
        },
        "CreateTime": 1603383451.0,
        "CreateTableDefaultPermissions": [
            {
                "Principal": {
                    "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                },
                "Permissions": [
                    "ALL"
                ]
            }
        ]
    }
}
```

Pour plus d'informations à ce sujet AWS CLI, consultez le guide de AWS Command Line Interface l'utilisateur.

CREATE TABLE

Permet de créer une table avec le nom et les paramètres que vous spécifiez.

> **Note**
>
> Cette page contient un résumé des informations de référence. Pour plus d'informations sur la création des tables dans Athena et un exemple de déclaration `CREATE TABLE`, consultez Création de tables dans Athena. En guise d'exemple en matière de création d'une base de données, de création d'une table et d'exécution d'une requête `SELECT` sur la table dans Athena, voir Mise en route.

Résumé

```
CREATE EXTERNAL TABLE [IF NOT EXISTS]
 [db_name.]table_name [(col_name data_type [COMMENT col_comment] [, ...] )]
 [COMMENT table_comment]
 [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
 [CLUSTERED BY (col_name, col_name, ...) INTO num_buckets BUCKETS]
 [ROW FORMAT row_format]
 [STORED AS file_format]
 [WITH SERDEPROPERTIES (...)]
 [LOCATION 's3://DOC-EXAMPLE-BUCKET/[folder]/']
 [TBLPROPERTIES ( ['has_encrypted_data'='true | false',]
 ['classification'='aws_glue_classification',] property_name=property_value [, ...] ) ]
```

Paramètres

EXTERNAL

Indique que la table est basée sur un fichier de données sous-jacent qui existe dans Simple Storage Service (Amazon S3), dans l'emplacement `LOCATION` que vous avez spécifié. Sauf lors de la création de tables Iceberg, utilisez toujours le mot-clé `EXTERNAL`. Si vous utilisez `CREATE TABLE` sans le mot clé `EXTERNAL` pour des tables non Iceberg, Athena émet une erreur. Lorsque vous créez une table externe, les données référencées doivent respecter le format par défaut ou le format que vous spécifiez à l'aide des clauses `ROW FORMAT`, `STORED AS` et `WITH SERDEPROPERTIES`.

[IF NOT EXISTS]

Ce paramètre vérifie si une table ayant le même nom existe déjà. Si c'est le cas, le paramètre renvoie TRUE et Amazon Athena annule l'action `CREATE TABLE`. L'annulation ayant lieu avant qu'Athéna n'appelle le catalogue de données, aucun événement n'est émis. AWS CloudTrail

[db_name.]table_name

Spécifie un nom pour la table à créer. Le paramètre facultatif `db_name` indique la base de données dans laquelle se trouve la table. Si ce paramètre n'est pas spécifié, la base de données en cours est utilisée par défaut. Si le nom de la table comporte des chiffres, mettez `table_name` entre guillemets, par exemple `"table123"`. Si `table_name` commence par un trait de soulignement, utilisez des accents graves, par exemple `` `_mytable` ``. Les caractères spéciaux (autres que le trait de soulignement) ne sont pas pris en charge.

Les noms de table Athena ne sont pas sensibles à la casse. Cependant, si vous travaillez avec Apache Spark, les noms de table doivent être en minuscule.

[(col_name data_type [COMMENT col_comment] [, ...])]

Indique le nom de chaque colonne à créer, ainsi que le type de données de la colonne. Les noms de colonne n'acceptent pas de caractères spéciaux autres que le trait de soulignement (`_`). Si `col_name` commence par un trait de soulignement, placez le nom de colonne entre des accents graves, par exemple `` `_mycolumn` ``.

`data_type` peut avoir l'une des valeurs suivantes :

- `boolean` : les valeurs sont `true` et `false`.
- `tinyint` – Un entier signé de 8 bits au format de complément à deux avec une valeur minimum de -2^7 et une valeur maximum de 2^7-1.
- `smallint` – Un entier signé de 16 bits au format de complément à deux avec une valeur minimum de -2^15 et une valeur maximum de 2^15-1.
- `int` – Dans les requêtes en langage de définition de données (DDL) telles que `CREATE TABLE`, utilisez le mot-clé `int` pour représenter un entier. Dans d'autres requêtes, utilisez le mot clé `integer`, où `integer` est représenté comme une valeur signée de 32 bits au format complément à deux, avec une valeur minimale de -2^31 et une valeur maximale de 2^31-1. Dans le pilote JDBC, `integer` est renvoyé, pour assurer la compatibilité avec les applications d'analyse d'entreprise.
- `bigint` – Un entier signé de 64 bits au format de complément à deux avec une valeur minimum de -2^63 et une valeur maximum de 2^63-1.

- `double` : un nombre à virgule flottante signé de 64 bits en double précision. La plage va de 4.94065645841246544e-324d à 1.79769313486231570e+308d, positif ou négatif. `double` suit le standard IEEE pour l'arithmétique à virgule flottante (IEEE 754).
- `float` : un nombre à virgule flottante signé à simple précision de 32 bits. La plage va de 1.40129846432481707e-45 à 3.40282346638528860e+38, positif ou négatif. `float` suit le standard IEEE pour l'arithmétique à virgule flottante (IEEE 754). Équivalent à `real` dans Presto. Dans Athena, utilisez `float` dans les Instructions DDL comme `CREATE TABLE` et `real` dans les fonctions SQL comme `SELECT CAST`. Le AWS Glue crawler renvoie des valeurs`float`, et Athena les `real` traduit `float` et les saisit en interne (voir 5 juin 2018 les notes de publication).
- `decimal [ (`*`precision`*`, `*`scale`*`) ]`, où *`precision`* est le nombre total de chiffres et *`scale`* (facultatif) est le nombre de chiffres dans la partie fraction, la valeur par défaut est 0. Par exemple, utilisez ces définitions de type : `decimal(11,5)`, `decimal(15)`. La valeur maximale pour la *`précision`* est de 38, et la valeur maximale pour l'*`échelle`* est de 38.

  Pour spécifier des valeurs décimales comme les littéraux, par exemple lorsque vous sélectionnez des lignes avec une valeur décimale dans une expression de requête DDL, spécifiez la définition de type `decimal` et répertoriez la valeur décimale en tant que valeur littérale (entre des guillemets simples) dans votre requête, comme dans l'exemple suivant : `decimal_value = decimal '0.12'`.
- `char` : données en caractères de longueur fixe, avec une longueur spécifiée entre 1 et 255, par exemple `char(10)`. Pour plus d'informations, consultez la section relative au type de données Hive CHAR.
- `varchar` : données en caractères de longueur variable, avec une longueur spécifiée entre 1 et 65535, par exemple `varchar(10)`. Pour plus d'informations, consultez la section relative au type de données Hive VARCHAR.
- `string` : libellé chaîne entre guillemets simples ou doubles.

  > **Note**
  >
  > Les types de données autres que les chaînes de caractères ne peuvent pas être convertis en `string` dans Athena ; convertissez-les plutôt en `varchar`.
- `binary` – (pour les données au format Parquet)

- `date` : date au format ISO, par exemple *YYYY*-*MM*-*DD*. Par exemple, `date '2008-09-15'`. Une exception est l'SerDeOpenCSV, qui utilise le nombre de jours écoulés depuis le 1er janvier 1970. Pour plus d'informations, consultez SerDe OpenCSV pour le traitement des fichiers CSV.
- `timestamp` : date et heure instantanée dans `java.sql.Timestamp` jusqu'à une résolution maximale de millisecondes, comme *yyyy*-*MM*-*dd HH*:*mm*:*ss*`[.f...]`. Par exemple, `timestamp '2008-09-15 03:04:05.324'`. L'SerDeOpenCSV constitue une exception, car il `TIMESTAMP` utilise des données au format numérique UNIX (par exemple,). `1579059880000` Pour plus d'informations, consultez SerDe OpenCSV pour le traitement des fichiers CSV.
- `array` < data_type >
- `map` < primitive_type, data_type >
- `struct` < col_name : data_type [comment col_comment] [, ...] >

[COMMENT table_comment]

Crée la propriété de table `comment` et la renseigne avec l'information `table_comment` que vous spécifiez.

[PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]

Crée une table partitionnée avec une ou plusieurs colonnes de partition dont les valeurs `col_name`, `data_type` et `col_comment` sont spécifiées. Une table peut comporter une ou plusieurs partitions, qui se composent d'une combinaison nom/valeur de colonne distincte. Un répertoire de données distinct est créé pour chaque combinaison spécifiée, ce qui peut améliorer les performances des requêtes dans certaines circonstances. Les colonnes partitionnées n'existent pas au sein même des données de la table. Si la valeur de `col_name` est identique à celle d'une colonne de table, une erreur est renvoyée. Pour en savoir plus, consultez Partitionnement de données.

> **Note**
>
> Une fois que vous avez créé une table avec des partitions, exécutez une requête ultérieure composée de la clause MSCK REPAIR TABLE pour actualiser les métadonnées de partition, par exemple `MSCK REPAIR TABLE cloudfront_logs;`. Pour les partitions qui ne sont pas compatibles avec Hive, utilisez ALTER TABLE ADD PARTITION pour charger les partitions de manière à pouvoir interroger les données.

[CLUSTERED BY (col_name, col_name, ...) INTO num_buckets BUCKETS]

Divise, avec ou sans partitionnement, les données des colonnes `col_name` spécifiées en sous-ensembles de données appelés compartiments. Le paramètre `num_buckets` indique le nombre de compartiments à créer. Le compartimentage peut améliorer les performances de certaines requêtes sur de grands jeux de données.

[ROW FORMAT row_format]

Spécifie le format de ligne de la table et, le cas échéant, de ses données source sous-jacentes. Pour `row_format`, vous pouvez spécifier un ou plusieurs délimiteurs avec la clause `DELIMITED`, ou utiliser la clause `SERDE` comme décrit ci-dessous. S'`ROW FORMAT`il est omis ou `ROW FORMAT DELIMITED` spécifié, un natif SerDe est utilisé.

- [DELIMITED FIELDS TERMINATED BY char [ESCAPED BY char]]
- [DELIMITED COLLECTION ITEMS TERMINATED BY char]
- [MAP KEYS TERMINATED BY char]
- [LINES TERMINATED BY char]
- [NULL DEFINED AS char]

  Disponible uniquement avec Hive 0.13 et lorsque le format de fichier STORED AS a pour valeur `TEXTFILE`.

--OR--

- SERDE 'serde_name' [WITH SERDEPROPERTIES ("property_name" = "property_value", "property_name" = "property_value" [, ...])]

  Le `serde_name` indique le SerDe à utiliser. La `WITH SERDEPROPERTIES` clause vous permet de fournir une ou plusieurs propriétés personnalisées autorisées par le SerDe.

[STORED AS format_fichier]

Spécifie le format de fichier pour les données de table. Si ce paramètre n'est pas spécifié, `TEXTFILE` est la valeur par défaut. Les options de `file_format` sont les suivantes :

- SEQUENCEFILE
- TEXTFILE
- RCFILE
- ORC

- PARQUET
- AVRO
- ION
- INPUTFORMAT nom_classe_format_entrée OUTPUTFORMAT nom_classe_format_sortie

[LIEU 's3://DOC-EXAMPLE-BUCKET/[folder]/']

Indique l'emplacement des données sous-jacentes dans Simple Storage Service (Amazon S3) à partir duquel la table est créée. Le chemin d'accès de l'emplacement doit être un nom de compartiment ou un nom de compartiment et un ou plusieurs dossiers. Si vous utilisez des partitions, spécifiez la racine des données partitionnées. Pour plus d'informations sur l'emplacement des tables, consultez Emplacement de table dans Simple Storage Service (Amazon S3). Pour obtenir des informations sur le format des données et les autorisations, consultez Exigences pour les tables dans Athena et les données dans Simple Storage Service (Amazon S3).

Utilisez une barre oblique pour votre dossier ou compartiment. N'utilisez pas de noms de fichiers ou de caractères généraux.

Utilisez :

`s3://DOC-EXAMPLE-BUCKET/`

`s3://DOC-EXAMPLE-BUCKET/folder/`

`s3://DOC-EXAMPLE-BUCKET/folder/anotherfolder/`

N'utilisez pas :

`s3://DOC-EXAMPLE-BUCKET`

`s3://DOC-EXAMPLE-BUCKET/*`

`s3://DOC-EXAMPLE-BUCKET/mydatafile.dat`

[TBLPROPERTIES (['has_encrypted_data'='true | false',] ['classification'='classification_value',] property_name=property_value [, ...])]

Spécifie des paires clés-valeurs de métadonnées personnalisées pour la définition de la table en plus des propriétés de table prédéfinies, par exemple `"comment"`.

has_encrypted_data : Athena dispose d'une propriété intégrée, `has_encrypted_data`. Attribuez la valeur `true` à cette propriété pour indiquer que l'ensemble de données sous-jacent spécifié par

`LOCATION` est chiffré. Si ce paramètre n'est pas spécifié et si les paramètres du groupe de travail ne remplacent pas les paramètres côté client, `false` est utilisé. Si ce paramètre n'est pas spécifié ou si la valeur `false` est définie lorsque des données sous-jacentes sont chiffrées, la requête génère une erreur. Pour plus d'informations, consultez Chiffrement au repos.

classification — Les tables créées pour Athena dans la CloudTrail console sont ajoutées en `cloudtrail` tant que valeur à la `classification` propriété. Pour exécuter des tâches AWS Glue ETL, vous devez créer une table avec la `classification` propriété indiquant le type de données pour AWS Glue as `csv`parquet,`orc`,`avro`, ou`json`. Par exemple, `'classification'='csv'`. Les tâches ETL échoueront si vous ne spécifiez pas cette propriété. Vous pouvez par conséquent la spécifier à l'aide de la console AWS Glue , de l'API ou de l'interface de ligne de commande (CLI). Pour plus d'informations, consultez Utiliser des AWS Glue jobs pour l'ETL avec Athena la section « Création de tâches dans AWS Glue » dans le manuel du AWS Glue développeur.

compression_level : la propriété `compression_level` spécifie le niveau de compression à utiliser. Cette propriété s'applique uniquement à la compression ZSTD. Les valeurs possibles sont comprises entre 1 et 22. La valeur par défaut est 3. Pour plus d'informations, consultez Utilisation des niveaux de compression ZSTD dans Athena.

Pour plus d'informations sur les autres propriétés de table, consultez ALTER TABLE SET TBLPROPERTIES.

Exemples

L'exemple d'`CREATE TABLE`instruction suivant crée une table basée sur des données planétaires séparées par des tabulations stockées dans Amazon S3.

```
CREATE EXTERNAL TABLE planet_data (
  planet_name string,
  order_from_sun int,
  au_to_sun float,
  mass float,
  gravity_earth float,
  orbit_years float,
  day_length float
  )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
```

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/tsv/'
```

Notez les points suivants :

- La `ROW FORMAT DELIMITED` clause indique que les données sont délimitées par un caractère spécifique.
- La `FIELDS TERMINATED BY '\t'` clause indique que les champs des données TSV sont séparés par le caractère de tabulation ('\ t').
- La `STORED AS TEXTFILE` clause indique que les données sont stockées sous forme de fichiers texte brut dans Amazon S3.

Pour interroger les données, vous pouvez utiliser une `SELECT` instruction simple comme celle-ci :

```
SELECT * FROM planet_data
```

Pour utiliser cet exemple pour créer votre propre table TSV dans Athena, remplacez les noms de table et de colonne par les noms et types de données de votre propre table et de vos colonnes, et mettez à jour `LOCATION` la clause pour qu'elle pointe vers le chemin Amazon S3 où vos fichiers TSV sont stockés.

Pour plus d'informations sur la création de tables, consultez Création de tables dans Athena.

CREATE TABLE AS

Crée une table remplie avec les résultats d'une requête SELECT. Pour créer une table vide, utilisez CREATE TABLE. `CREATE TABLE AS` combine une instruction DDL `CREATE TABLE` avec une instruction DML `SELECT` et contient donc techniquement tant des instructions DDL que DML. Notez que, bien que l'instruction `CREATE TABLE AS` soit regroupée ici avec d'autres instructions DDL, les requêtes CTAS dans Athena sont traitées comme des requêtes DML à des fins de Service Quotas. Pour plus d'informations sur les Service Quotas Athena, consultez Service Quotas.

> **Note**
>
> Pour les instructions CTAS, le paramètre de propriétaire du compartiment attendu ne s'applique pas à l'emplacement de la table de destination dans Simple Storage Service (Amazon S3). Le paramètre de propriétaire du compartiment attendu s'applique uniquement à l'emplacement de sortie Simple Storage Service (Amazon S3) que vous spécifiez pour

les résultats de la requête Athena. Pour plus d'informations, consultez Spécification d'un emplacement de résultats de requête à l'aide de la console Athena.

Pour plus d'informations sur l'instruction `CREATE TABLE AS` allant au-delà de la portée de cette rubrique de référence, consultez Création d'une table à partir des résultats des requêtes (CTAS).

Rubriques

Résumé

```
CREATE TABLE table_name
[ WITH ( property_name = expression [, ...] ) ]
AS query
[ WITH [ NO ] DATA ]
```

Où :

WITH (property_name = expression [, ...])

Liste de propriétés facultatives de la table CTAS, dont certaines sont spécifiques au format de stockage de données. veuillez consulter Propriétés de la table CTAS.

query

Une requête SELECT qui est utilisée pour créer une nouvelle table.

> **Important**
>
> Si vous envisagez de créer une requête avec les partitions, spécifiez les noms des colonnes partitionnées à la fin de la liste de colonnes dans l'instruction `SELECT`.

[WITH [NO] DATA]

Si `WITH NO DATA` est utilisé, une nouvelle table vide avec le même schéma que la table d'origine est créée.

> **Note**
>
> Pour inclure des en-têtes de colonne dans la sortie du résultat de votre requête, vous pouvez utiliser une simple requête `SELECT` au lieu d'une requête CTAS. Vous pouvez récupérer les résultats depuis l'emplacement des résultats de votre requête ou télécharger les résultats directement à l'aide de la console Athena. Pour plus d'informations, consultez Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie.

Propriétés de la table CTAS

Chaque table CTAS dans Athena dispose d'une liste des propriétés facultatives de table CTAS que vous spécifiez à l'aide de `WITH (property_name = expression [, ...] )`. Pour obtenir des informations sur l'utilisation de ces paramètres, consultez Exemples de requêtes CTAS.

`WITH (property_name = expression [, ...], )`

`table_type = ['HIVE', 'ICEBERG']`

Facultatif. L'argument par défaut est `HIVE`. Spécifie le type de table de la table résultante

Exemple :

```
WITH (table_type ='ICEBERG')
```

`external_location = [location]`

> **Note**
>
> Comme les tables Iceberg ne sont pas externes, cette propriété ne s'applique pas aux tables Iceberg. Pour définir l'emplacement de la racine d'une table Iceberg dans une instruction CTAS, utilisez la propriété `location` décrite plus loin dans cette rubrique.

Facultatif. L'emplacement où Athena enregistre votre requête CTAS dans Amazon S3.

Exemple :

```
WITH (external_location ='s3://DOC-EXAMPLE-BUCKET/tables/parquet_table/')
```

Athena n'utilise pas le même chemin pour les résultats de la requête deux fois. Si vous spécifiez l'emplacement manuellement, assurez-vous que cet emplacement Simple Storage Service (Amazon S3) n'a pas de données. Athena ne tente jamais de supprimer vos données. Si vous souhaitez utiliser le même emplacement à nouveau, nettoyez manuellement les données, faute de quoi votre requête CTAS échouera.

Si vous exécutez une requête CTAS qui spécifie un `external_location` dans un groupe de travail qui applique un emplacement de résultats de requête, la requête échoue avec un message d'erreur. Pour voir l'emplacement des résultats de la requête spécifié pour le groupe de travail, consultez les détails du groupe de travail.

Si votre groupe de travail remplace le paramètre côté client pour l'emplacement des résultats de requête, Athena crée votre table à l'emplacement suivant :

```
s3://DOC-EXAMPLE-BUCKET/tables/query-id/
```

Si vous n'utilisez pas la propriété `external_location` pour spécifier un emplacement et que votre groupe de travail ne remplace pas les paramètres côté client, Athena utilise votre paramètre côté client pour l'emplacement des résultats de la requête afin de créer votre table à l'emplacement suivant :

```
s3://DOC-EXAMPLE-BUCKET/Unsaved-or-query-name/year/month/date/tables/query-id/
```

`is_external = [boolean]`

Facultatif. Indique si la table est une table externe. Par défaut, la valeur est true. Pour les tables Iceberg, cette valeur doit être définie sur false.

Exemple :

```
WITH (is_external = false)
```

`location = [location]`

Nécessaire pour les tables Iceberg. Spécifie l'emplacement racine de la table Iceberg à créer à partir des résultats de la requête.

Exemple :

```
WITH (location ='s3://DOC-EXAMPLE-BUCKET/tables/iceberg_table/')
```

field_delimiter = [delimiter]

Facultatif et spécifique aux formats de stockage de données à base de texte. Délimiteur de champ à un seul caractère pour les fichiers CSV, TSV et les fichiers texte. Par exemple, `WITH (field_delimiter = ',')`. Actuellement, les délimiteurs de champ à plusieurs caractères ne sont pas pris en charge pour les requêtes CTAS. Si vous ne spécifiez pas de délimiteur de champs, `\001` est utilisé par défaut.

format = [storage_format]

Format de stockage pour les résultats de requête CTAS, tel que `ORC`, `PARQUET`, `AVRO`, `JSON`, `ION` ou `TEXTFILE`. Pour les tables Iceberg, les formats autorisés sont `ORC`, `PARQUET` et `AVRO`. En cas d'omission, `PARQUET` est utilisé par défaut. Le nom de ce paramètre, `format`, doit être répertorié en minuscules, faute de quoi votre requête CTAS échoue.

Exemple :

```
WITH (format = 'PARQUET')
```

bucketed_by = ARRAY[column_name[,…], bucket_count = [int]]

> **Note**
>
> Cette propriété ne s'applique pas aux tables Iceberg. Pour les tables Iceberg, utilisez le partitionnement avec la transformation des compartiments.

Tableau de compartiments pour la mise en compartiments des données. En cas d'omission, Athena ne met pas vos données en compartiments dans cette requête.

bucket_count = [int]

> **Note**
>
> Cette propriété ne s'applique pas aux tables Iceberg. Pour les tables Iceberg, utilisez le partitionnement avec la transformation des compartiments.

Nombre de compartiments pour la mise en compartiments de vos données. En cas d'omission, Athena ne met pas vos données en compartiments. Exemple :

```
CREATE TABLE bucketed_table WITH (
  bucketed_by = ARRAY[column_name],
  bucket_count = 30, format = 'PARQUET',
  external_location ='s3://DOC-EXAMPLE-BUCKET/tables/parquet_table/'
) AS
SELECT
  *
FROM
  table_name
```

partitioned_by = ARRAY[col_name[,…]]

> **Note**
>
> Cette propriété ne s'applique pas aux tables Iceberg. Pour utiliser les transformations de partition pour les tables Iceberg, utilisez la propriété `partitioning` décrite plus loin dans cette rubrique.

Facultatif. Tableau composé de colonnes à l'aide duquel la table CTAS est partitionnée. Vérifiez que les noms des colonnes partitionnées sont répertoriés à la fin de la liste des colonnes dans l'instruction `SELECT`.

partitioning = ARRAY[partition_transform, ...]

Facultatif. Spécifie le partitionnement de la table Iceberg à créer. Iceberg prend en charge une grande variété de transformations et d'évolutions de partitions. Les transformations de partition sont résumées dans le tableau suivant.

Transformation	Description
`year(ts)`	Crée une partition pour chaque année. La valeur de la partition est la différence entière en années entre `ts` et le 1er janvier 1970.
`month(ts)`	Crée une partition pour chaque mois de chaque année. La valeur de la partition est la différence entière en mois entre `ts` et le 1er janvier 1970.

Transformation	Description
`day(ts)`	Crée une partition pour chaque jour de chaque année. La valeur de la partition est la différence entière en jours entre `ts` et le 1er janvier 1970.
`hour(ts)`	Crée une partition pour chaque heure de chaque jour. La valeur de la partition est un horodatage dont les minutes et les secondes sont définies sur zéro.
`bucket(x, nbuckets)`	Hache les données dans le nombre de compartiments spécifié. La valeur de la partition est un hachage entier de `x`, avec une valeur comprise entre 0 et `nbuckets - 1`, inclus.
`truncate(s, nchars)`	Fait des premiers caractères `nchars` de `s` la valeur de la partition.

Exemple :

```
WITH (partitioning = ARRAY['month(order_date)',
                           'bucket(account_number, 10)',
                           'country']))
```

`optimize_rewrite_min_data_file_size_bytes = [long]`

Facultatif. Configuration spécifique à l'optimisation des données. Les fichiers inférieurs à la valeur spécifiée sont inclus pour l'optimisation. La valeur par défaut est de 0,75 fois la valeur de `write_target_data_file_size_bytes`. Cette propriété ne s'applique qu'aux tables Iceberg. Pour plus d'informations, consultez Optimisation des tables Iceberg.

Exemple :

```
WITH (optimize_rewrite_min_data_file_size_bytes = 402653184)
```

`optimize_rewrite_max_data_file_size_bytes = [long]`

Facultatif. Configuration spécifique à l'optimisation des données. Les fichiers supérieurs à la valeur spécifiée sont inclus pour l'optimisation. La valeur par défaut est de 1,8 fois la

valeur de `write_target_data_file_size_bytes`. Cette propriété ne s'applique qu'aux tables Iceberg. Pour plus d'informations, consultez Optimisation des tables Iceberg.

Exemple :

```
WITH (optimize_rewrite_max_data_file_size_bytes = 966367641)
```

`optimize_rewrite_data_file_threshold = [int]`

Facultatif. Configuration spécifique à l'optimisation des données. S'il y a moins de fichiers de données nécessitant une optimisation que le seuil donné, les fichiers ne sont pas réécrits. Cela permet d'accumuler un plus grand nombre de fichiers de données afin de produire des fichiers plus proches de la taille cible et de sauter les calculs inutiles afin de réduire les coûts. La valeur par défaut est 5. Cette propriété ne s'applique qu'aux tables Iceberg. Pour plus d'informations, consultez Optimisation des tables Iceberg.

Exemple :

```
WITH (optimize_rewrite_data_file_threshold = 5)
```

`optimize_rewrite_delete_file_threshold = [int]`

Facultatif. Configuration spécifique à l'optimisation des données. S'il y a moins de fichiers de suppression associés à un fichier de données que le seuil, le fichier de données n'est pas réécrit. Cela permet d'accumuler un plus grand nombre de fichiers de suppression pour chaque fichier de données afin de réduire les coûts. La valeur par défaut est 2. Cette propriété ne s'applique qu'aux tables Iceberg. Pour plus d'informations, consultez Optimisation des tables Iceberg.

Exemple :

```
WITH (optimize_rewrite_delete_file_threshold = 2)
```

`vacuum_min_snapshots_to_keep = [int]`

Facultatif. Configuration spécifique à Vacuum. Le nombre minimum d'instantanés les plus récents à retenir. La valeur par défaut est 1. Cette propriété ne s'applique qu'aux tables Iceberg. Pour plus d'informations, consultez VACUUM.

> **Note**
>
> La propriété `vacuum_min_snapshots_to_keep` nécessite la version 3 du moteur Athena.

Exemple :

```
WITH (vacuum_min_snapshots_to_keep = 1)
```

`vacuum_max_snapshot_age_seconds = [long]`

Facultatif. Configuration spécifique à Vacuum. Période en secondes qui représente l'âge des instantanés à retenir. La valeur par défaut est de 432 000 (5 jours). Cette propriété ne s'applique qu'aux tables Iceberg. Pour plus d'informations, consultez VACUUM.

> **Note**
>
> La propriété `vacuum_max_snapshot_age_seconds` nécessite la version 3 du moteur Athena.

Exemple :

```
WITH (vacuum_max_snapshot_age_seconds = 432000)
```

`write_compression = [compression_format]`

Type de compression à utiliser pour n'importe quel format de stockage permettant de spécifier la compression. La valeur `compression_format` spécifie la compression à utiliser lorsque les données sont écrites dans la table. Vous pouvez spécifier une compression pour les formats de fichiers `TEXTFILE`, `JSON`, `PARQUET` et `ORC`.

Par exemple, si la propriété `format` spécifie `PARQUET` comme format de stockage, la valeur de `write_compression` spécifie le format de compression pour Parquet. Dans ce cas, spécifier une valeur pour `write_compression` revient à spécifier une valeur pour `parquet_compression`.

Par exemple, si la propriété `format` spécifie `ORC` comme format de stockage, la valeur de `write_compression` spécifie le format de compression pour ORC. Dans ce cas,

spécifier une valeur pour `write_compression` revient à spécifier une valeur pour `orc_compression`.

Il n'est pas possible de spécifier plusieurs propriétés de table de format de compression dans la même requête CTAS. Par exemple, vous ne pouvez pas spécifier les `write_compression` et `parquet_compression` simultanément dans la même requête. Il en va de même pour `write_compression` et `orc_compression`. Pour plus d'informations sur les formats de compression pris en charge par chaque format de fichier, consultez Prise en charge de la compression Athena.

`orc_compression = [compression_format]`

Type de compression à utiliser pour le format de fichier `ORC` lorsque les données `ORC` sont écrites dans la table. Par exemple, `WITH (orc_compression = 'ZLIB')`. Les morceaux du fichier `ORC` (à l'exception du fichier `ORC` Postscript) sont compressés en utilisant la compression que vous avez spécifiée. En cas d'omission, la compression ZLIB est utilisée par défaut pour `ORC`.

> **Note**
>
> Pour des raisons de cohérence, nous vous recommandons d'utiliser la propriété `write_compression` au lieu de `orc_compression`. Utilisez la propriété `format` pour spécifier le format de stockage comme`ORC`, puis utilisez la propriété `write_compression` pour spécifier le format de compression que `ORC` utilisera.

`parquet_compression = [compression_format]`

Type de compression à utiliser pour le format de fichier Parquet lorsque les données Parquet sont écrites dans la table. Par exemple, `WITH (parquet_compression = 'SNAPPY')`. Cette compression est appliquée aux blocs de colonnes dans les fichiers Parquet. En cas d'omission, la compression GZIP est utilisée par défaut pour Parquet.

> **Note**
>
> Pour des raisons de cohérence, nous vous recommandons d'utiliser la propriété `write_compression` au lieu de `parquet_compression`. Utilisez la propriété `format` pour spécifier le format de stockage comme`PARQUET`, puis utilisez la propriété `write_compression` pour spécifier le format de compression que `PARQUET` utilisera.

`compression_level = [compression_level]`

Le niveau de compression à utiliser. Cette propriété s'applique uniquement à la compression ZSTD. Les valeurs possibles sont comprises entre 1 et 22. La valeur par défaut est 3. Pour plus d'informations, consultez Utilisation des niveaux de compression ZSTD dans Athena.

Exemples

Pour obtenir des exemples de requêtes CTAS, consultez les ressources suivantes.

- Exemples de requêtes CTAS
- Utilisation de CTAS et INSERT INTO pour ETL et l'analyse des données
- Utilisation des instructions CTAS avec Amazon Athena pour réduire les coûts et améliorer les performances
- Utilisation de CTAS et de INSERT INTO pour contourner la limite de 100 partitions

CREATE VIEW

Crée une nouvelle vue à partir d'une requête `SELECT` spécifiée. La vue est une table logique qui peut être référencée par de futures requêtes. Les vues ne contiennent pas de données et n'écrivent pas de données. Au lieu de cela, la requête spécifiée par la vue s'exécute chaque fois que vous référencez la vue par une autre requête.

> **Note**
>
> Cette rubrique fournit des informations récapitulatives à titre de référence. Pour plus d'informations sur l'utilisation des vues dans Athena, voir Utilisation des vues. Pour plus d'informations sur les limites d'affichage, consultezLimitations pour les vues.

Résumé

```
CREATE [ OR REPLACE ] VIEW view_name AS query
```

La clause `OR REPLACE` en option vous permet de mettre à jour la vue existante en la remplaçant. Pour plus d'informations, consultez Création de vues.

Exemples

Pour créer une vue `test` à partir de la table `orders`, utilisez une requête semblable à la suivante :

```
CREATE VIEW test AS
SELECT
orderkey,
orderstatus,
totalprice / 2 AS half
FROM orders;
```

Pour créer une vue `orders_by_date` à partir de la table `orders`, utilisez la requête suivante :

```
CREATE VIEW orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
```

Pour mettre à jour une vue existante, utilisez un exemple semblable à ce qui suit :

```
CREATE OR REPLACE VIEW test AS
SELECT orderkey, orderstatus, totalprice / 4 AS quarter
FROM orders;
```

Consultez aussi SHOW COLUMNS, SHOW CREATE VIEW, DESCRIBE VIEW et DROP VIEW.

DESCRIBE

Affiche une ou plusieurs colonnes, y compris les colonnes de partition, pour la table spécifiée. Cette commande est utile pour examiner les attributs de colonnes complexes.

Résumé

```
DESCRIBE [EXTENDED | FORMATTED] [db_name.]table_name [PARTITION partition_spec]
 [col_name ( [.field_name] | [.'$elem$'] | [.'$key$'] | [.'$value$'] )]
```

> **Important**
>
> La syntaxe de cette instruction est `DESCRIBE table_name`, et non `DESCRIBE TABLE table_name`. L'utilisation de cette dernière syntaxe entraîne le message d'erreur SemanticException FAILED : [Error 10001] : Table introuvable.

Paramètres

[EXTENDED | FORMATTED]

Détermine le format de la sortie. L'omission de ces paramètres affiche les noms de colonnes et leurs types de données correspondants, y compris les colonnes de partition, sous forme de tableau. La spécification de `FORMATTED` affiche non seulement les noms de colonnes et les types de données sous forme de tableau, mais également des informations détaillées sur le tableau et le stockage. `EXTENDED` affiche les informations sur les colonnes et les types de données sous forme de tableau, ainsi que les métadonnées détaillées de la table sous la forme sérialisée Thrift. Ce format est moins lisible et principalement utile pour le débogage.

[PARTITION partition_spec]

Si elle est incluse, elle répertorie les métadonnées de la partition spécifiée par `partition_spec`, où `partition_spec` est au format `(partition_column = partition_col_value, partition_column = partition_col_value, ...)`.

[col_name ([.field_name] | [.'$elem$'] | [.'key'] | [.'$value$'])*]

Spécifie la colonne et les attributs à examiner. Vous pouvez spécifier `.field_name` pour l'élément d'une struct, `'$elem$'` pour l'élément d'un tableau, `'$key$'` pour une clé de mappage et `'$value$'` pour une valeur de mappage. Vous pouvez le faire de façon récursive pour explorer plus avant la colonne complexe.

Exemples

```
DESCRIBE orders
```

```
DESCRIBE FORMATTED mydatabase.mytable PARTITION (part_col = 100) columnA;
```

La requête et la sortie suivantes affichent des informations sur les colonnes et les types de données provenant d'une table `impressions` qui se base sur des exemples de données Amazon EMR.

```
DESCRIBE impressions
```

```
requestbegintime          string                                              from
 deserializer
adid                      string                                              from
 deserializer
impressionid              string                                              from
 deserializer
referrer                  string                                              from
 deserializer
useragent                 string                                              from
 deserializer
usercookie                string                                              from
 deserializer
ip                        string                                              from
 deserializer
number                    string                                              from
 deserializer
processid                 string                                              from
 deserializer
browsercokie              string                                              from
 deserializer
requestendtime            string                                              from
 deserializer
timers                    struct<modellookup:string,requesttime:string>  from
 deserializer
threadid                  string                                              from
 deserializer
hostname                  string                                              from
 deserializer
sessionid                 string                                              from
 deserializer
dt                        string

# Partition Information
# col_name                data_type                comment

dt                        string
```

L'exemple de requête et de sortie suivant montre le résultat d'une même table lorsque l'option `FORMATTED` est utilisée.

```
DESCRIBE FORMATTED impressions
```

```
requestbegintime          string                                              from
 deserializer
adid                      string                                              from
 deserializer
impressionid              string                                              from
 deserializer
referrer                  string                                              from
 deserializer
useragent                 string                                              from
 deserializer
usercookie                string                                              from
 deserializer
ip                        string                                              from
 deserializer
number                    string                                              from
 deserializer
processid                 string                                              from
 deserializer
browsercokie              string                                              from
 deserializer
requestendtime            string                                              from
 deserializer
timers                    struct<modellookup:string,requesttime:string>  from
 deserializer
threadid                  string                                              from
 deserializer
hostname                  string                                              from
 deserializer
sessionid                 string                                              from
 deserializer
dt                        string

# Partition Information
# col_name                data_type                comment

dt                        string

# Detailed Table Information
Database:                 sampledb
Owner:                    hadoop
CreateTime:               Thu Apr 23 02:55:21 UTC 2020
```

```
LastAccessTime:           UNKNOWN
Protect Mode:             None
Retention:                0
Location:                 s3://us-east-1.elasticmapreduce/samples/hive-ads/tables/
impressions
Table Type:               EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                    TRUE
        transient_lastDdlTime     1587610521

# Storage Information
SerDe Library:                        org.openx.data.jsonserde.JsonSerDe
InputFormat:                          org.apache.hadoop.mapred.TextInputFormat
OutputFormat:
 org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat
Compressed:                           No
Num Buckets:                          -1
Bucket Columns:                       []
Sort Columns:                         []
Storage Desc Params:
        paths                                 requestbegintime, adid, impressionid,
 referrer, useragent, usercookie, ip
        serialization.format                  1
```

L'exemple de requête et de sortie suivant montre le résultat d'une même table lorsque l'option EXTENDED est utilisée. Les informations détaillées du tableau sont affichées sur une seule ligne, mais elles sont mises en forme ici pour plus de lisibilité.

```
DESCRIBE EXTENDED impressions
```

```
requestbegintime          string                                              from
 deserializer
adid                      string                                              from
 deserializer
impressionid              string                                              from
 deserializer
referrer                  string                                              from
 deserializer
useragent                 string                                              from
 deserializer
usercookie                string                                              from
 deserializer
```

```
ip                      string                                          from
 deserializer
number                  string                                          from
 deserializer
processid               string                                          from
 deserializer
browsercokie            string                                          from
 deserializer
requestendtime          string                                          from
 deserializer
timers                  struct<modellookup:string,requesttime:string>  from
 deserializer
threadid                string                                          from
 deserializer
hostname                string                                          from
 deserializer
sessionid               string                                          from
 deserializer
dt                      string

# Partition Information
# col_name              data_type               comment

dt                      string

Detailed Table Information      Table(tableName:impressions, dbName:sampledb,
 owner:hadoop, createTime:1587610521,
lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:
[FieldSchema(name:requestbegintime, type:string, comment:null),
FieldSchema(name:adid, type:string, comment:null), FieldSchema(name:impressionid,
 type:string, comment:null),
FieldSchema(name:referrer, type:string, comment:null), FieldSchema(name:useragent,
 type:string, comment:null),
FieldSchema(name:usercookie, type:string, comment:null), FieldSchema(name:ip,
 type:string, comment:null),
FieldSchema(name:number, type:string, comment:null), FieldSchema(name:processid,
 type:string, comment:null),
FieldSchema(name:browsercokie, type:string, comment:null),
 FieldSchema(name:requestendtime, type:string, comment:null),
FieldSchema(name:timers, type:struct<modellookup:string,requesttime:string>,
 comment:null), FieldSchema(name:threadid,
type:string, comment:null), FieldSchema(name:hostname, type:string, comment:null),
 FieldSchema(name:sessionid,
```

```
type:string, comment:null)], location:s3://us-east-1.elasticmapreduce/samples/hive-ads/
tables/impressions,
inputFormat:org.apache.hadoop.mapred.TextInputFormat,
outputFormat:org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat, compressed:false,
 numBuckets:-1,
serdeInfo:SerDeInfo(name:null, serializationLib:org.openx.data.jsonserde.JsonSerDe,
 parameters:{serialization.format=1,
paths=requestbegintime, adid, impressionid, referrer, useragent, usercookie, ip}),
 bucketCols:[], sortCols:[], parameters:{},
skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[],
 skewedColValueLocationMaps:{}),
storedAsSubDirectories:false), partitionKeys:[FieldSchema(name:dt, type:string,
 comment:null)],
parameters:{EXTERNAL=TRUE, transient_lastDdlTime=1587610521}, viewOriginalText:null,
 viewExpandedText:null,
tableType:EXTERNAL_TABLE)
```

DESCRIBE VIEW

Affiche la liste des colonnes pour la vue nommée. Vous pouvez ainsi examiner les attributs d'une vue complexe.

Résumé

```
DESCRIBE [db_name.]view_name
```

Exemple

```
DESCRIBE orders;
```

Consultez aussi SHOW COLUMNS, SHOW CREATE VIEW, SHOW VIEWS et DROP VIEW.

DROP DATABASE

Supprime la base de données nommée du catalogue. Si la base de données contient des tables, vous devez supprimer les tables avant d'exécuter `DROP DATABASE` ou utiliser la clause `CASCADE`. Les éléments `DATABASE` et `SCHEMA` sont interchangeables. Ils ont la même signification.

Résumé

```
DROP {DATABASE | SCHEMA} [IF EXISTS] database_name [RESTRICT | CASCADE]
```

Paramètres

[IF EXISTS]

Entraîne la suppression de l'erreur si `database_name` n'existe pas.

[RESTRICT|CASCADE]

Détermine la manière dont les tables de `database_name` sont considérées lors de l'opération `DROP`. Si vous spécifiez `RESTRICT`, la base de données n'est pas supprimée si elle contient des tables. Il s'agit du comportement de par défaut. Si vous spécifiez `CASCADE`, la base de données et toutes ses tables sont supprimées.

Exemples

```
DROP DATABASE clickstreams;
```

```
DROP SCHEMA IF EXISTS clickstreams CASCADE;
```

> **Note**
>
> Lorsque vous essayez de supprimer une base de données dont le nom comporte des caractères spéciaux (par exemple, `my-database`), vous pouvez recevoir un message d'erreur. Pour résoudre ce problème, essayez d'entourer le nom de la base de données de guillemets inversés (`). Pour plus d'informations sur la dénomination des bases de données dans Athena, veuillez consulter la rubrique Noms des tables, des bases de données et des colonnes.

DROP TABLE

Supprime la définition de table de métadonnées pour la table nommée `table_name`. Lorsque vous supprimez une table externe, les données sous-jacentes restent intactes.

Résumé

```
DROP TABLE [IF EXISTS] table_name
```

Paramètres

[IF EXISTS]

Entraîne la suppression de l'erreur si `table_name` n'existe pas.

Exemples

```
DROP TABLE fulfilled_orders
```

```
DROP TABLE IF EXISTS fulfilled_orders
```

Lorsque vous utilisez l'éditeur de requête de la console Athena pour supprimer une table contenant des caractères spéciaux autres que le trait de soulignement (_), utilisez des guillemets simples inversés, comme dans l'exemple suivant.

```
DROP TABLE `my-athena-database-01.my-athena-table`
```

Lorsque vous utilisez le connecteur JDBC pour supprimer une table contenant des caractères spéciaux, les guillemets inversés ne sont pas requis.

```
DROP TABLE my-athena-database-01.my-athena-table
```

DROP VIEW

Abandonne (supprime) une vue existante. La clause `IF EXISTS` en option génère l'erreur à supprimer si la vue n'existe pas.

Pour plus d'informations, consultez Utilisation des vues.

Résumé

```
DROP VIEW [ IF EXISTS ] view_name
```

Exemples

```
DROP VIEW orders_by_date
```

```
DROP VIEW IF EXISTS orders_by_date
```

Consultez aussi CREATE VIEW, SHOW COLUMNS, SHOW CREATE VIEW, SHOW VIEWS et DESCRIBE VIEW.

MSCK REPAIR TABLE

Utilisez la commande `MSCK REPAIR TABLE` pour mettre à jour les métadonnées dans le catalogue après avoir ajouté des partitions compatibles Hive.

La commande `MSCK REPAIR TABLE` analyse un système de fichiers tel que Simple Storage Service (Amazon S3) à la recherche de partitions compatibles avec Hive qui ont été ajoutées au système de fichiers après la création de la table. `MSCK REPAIR TABLE` compare les partitions dans les métadonnées de la table et les partitions dans S3. Si de nouvelles partitions sont présentes dans l'emplacement S3 que vous avez spécifié lors de la création de la table, elle ajoute ces partitions aux métadonnées et à la table Athena.

Lorsque vous ajoutez des partitions physiques, les métadonnées du catalogue deviennent incompatibles avec la disposition des données dans le système de fichiers et des informations sur les nouvelles partitions doivent être ajoutées au catalogue. Pour mettre à jour les métadonnées, exécutez `MSCK REPAIR TABLE` afin de pouvoir interroger les données des nouvelles partitions à partir d'Athena.

> **Note**
>
> La commande `MSCK REPAIR TABLE` ajoute uniquement des partitions aux métadonnées ; elle ne les supprime pas. Pour supprimer des partitions des métadonnées après la suppression manuelle des partitions dans Simple Storage Service (Amazon S3), exécutez la commande `ALTER TABLE table-name DROP PARTITION`. Pour plus d'informations, voir ALTER TABLE DROP PARTITION.

Considérations et restrictions

Gardez les points suivants à l'esprit lorsque vous utilisez `MSCK REPAIR TABLE` :

- Il est possible que l'ajout de toutes les partitions prenne un certain temps. Si cette opération expire, elle se retrouve dans un état incomplet où seules quelques partitions sont ajoutées au catalogue.

Vous devez exécuter `MSCK REPAIR TABLE` sur la même table jusqu'à ce que toutes les partitions aient été ajoutées. Pour plus d'informations, consultez Partitionnement de données dans Athena.

- Pour les partitions qui ne sont pas compatibles avec Hive, utilisez ALTER TABLE ADD PARTITION pour charger les partitions de manière à pouvoir interroger leurs données.
- Les emplacements de partition à utiliser avec Athena doivent utiliser le protocole `s3` (par exemple, `s3://DOC-EXAMPLE-BUCKET/folder/`). Dans Athena, les emplacements qui utilisent d'autres protocoles (par exemple, `s3a://bucket/folder/`) provoquent des échecs de requête lorsque les requêtes `MSCK REPAIR TABLE` sont exécutées sur les tables contenant les données.
- Étant donné que `MSCK REPAIR TABLE` analyse à la fois un dossier et ses sous-dossiers pour trouver un schéma de partition correspondant, veillez à conserver les données des différentes tables dans des hiérarchies de dossiers distinctes. Supposons, par exemple, que vous ayez des données pour le tableau 1 dans `s3://DOC-EXAMPLE-BUCKET1` et des données pour le tableau 2 dans`s3://DOC-EXAMPLE-BUCKET1/table-2-data`. Si les deux tables sont partitionnées par chaîne, `MSCK REPAIR TABLE` ajoutera les partitions de la table 2 à la table 1. Pour éviter cela, utilisez des structures de dossiers distinctes telles que `s3://DOC-EXAMPLE-BUCKET1` et à la `s3://DOC-EXAMPLE-BUCKET2` place. Notez que ce comportement est compatible avec Amazon EMR et Apache Hive.
- En raison d'un problème connu, `MSCK REPAIR TABLE` échoue silencieusement lorsque les valeurs de partition contiennent un caractère deux-points (`:`) (par exemple, lorsque la valeur de partition est un horodatage). Comme solution de contournement, utilisez ALTER TABLE ADD PARTITION.
- `MSCK REPAIR TABLE` n'ajoute pas de noms de colonnes de partition commençant par un trait de soulignement (_). Pour contourner cette limite, utilisez ALTER TABLE ADD PARTITION.

Résumé

```
MSCK REPAIR TABLE table_name
```

Exemples

```
MSCK REPAIR TABLE orders;
```

Résolution des problèmes

Après avoir exécuté`MSCK REPAIR TABLE`, si Athena n'ajoute pas les partitions au tableau dans le AWS Glue Data Catalog, vérifiez les points suivants :

- AWS Glue accès — Assurez-vous que le rôle AWS Identity and Access Management (IAM) dispose d'une politique autorisant l'`glue:BatchCreatePartition`action. Pour plus d'informations, consultez Autoriser la colle : BatchCreatePartition dans la politique IAM dans la suite de ce document.
- Accès Amazon S3 : assurez-vous que le rôle dispose d'une politique avec des autorisations suffisantes pour accéder à Amazon S3, y compris l'action `s3:DescribeJob`. Pour un exemple des actions Simple Storage Service (Amazon S3) à autoriser, voir l'exemple de politique de compartiment dans Accès inter-comptes aux compartiments Simple Storage Service (Amazon S3) dans Athena.
- Casses des clés d'objet Amazon S3 : assurez-vous que le chemin Amazon S3 est en minuscules et non en casse mixte (par exemple, `userid` au lieu de `userId`), ou utilisez `ALTER TABLE ADD PARTITION` pour spécifier les noms des clés d'objet. Pour plus d'informations, consultez Modifier ou redéfinir le chemin Amazon S3 dans la suite de ce document.
- Délais d'expiration des requêtes – La commande `MSCK REPAIR TABLE` est mieux utilisée lors de la création d'une table pour la première fois ou lorsqu'il existe une incertitude quant à la parité entre les données et les métadonnées de partition. Si vous utilisez `MSCK REPAIR TABLE` pour ajouter de nouvelles partitions fréquemment (par exemple, sur une base quotidienne) et que vous rencontrez des délais d'expiration des requêtes, envisagez d'utiliser ALTER TABLE ADD PARTITION.
- Partitions manquantes dans le système de fichiers : si vous supprimez manuellement une partition dans Amazon S3, puis exécutez `MSCK REPAIR TABLE`, il est possible que vous receviez le message d'erreur Partitions manquantes dans le système de fichiers. Cela se produit parce que la commande `MSCK REPAIR TABLE` ne supprime pas les partitions périmées des métadonnées de la table. Pour supprimer les partitions supprimées des métadonnées de la table, exécutez ALTER TABLE DROP PARTITION à la place. Notez que SHOW PARTITIONS (AFFICHER LES PARTITIONS) répertorie de manière similaire uniquement les partitions dans les métadonnées, pas les partitions dans le système de fichiers.
- Erreur « NullPointerException name is null »

  Si vous utilisez l'opération AWS Glue CreateTableAPI ou le AWS CloudFormation `AWS::Glue::Table`modèle pour créer une table à utiliser dans Athena sans spécifier la `TableType` propriété, puis si vous exécutez une requête DDL du type `SHOW CREATE TABLE` ou`MSCK REPAIR TABLE`, vous pouvez recevoir le message d'erreur FAILED : NullPointerException Name is null.

Pour résoudre l'erreur, spécifiez une valeur pour l'[TableInput](#)`TableType`attribut dans le cadre de l'appel ou du [AWS CloudFormation modèle](#) d' AWS Glue `CreateTable`API. Parmi les valeurs possibles pour `TableType` figurent `EXTERNAL_TABLE` ou `VIRTUAL_VIEW`.

Cette exigence s'applique uniquement lorsque vous créez une table à l'aide de l'opération AWS Glue `CreateTable` API ou du `AWS::Glue::Table` modèle. Si vous créez une table pour Athena en utilisant à l'aide d'une instruction DDL ou d'un Crawler AWS Glue , la propriété `TableType` est définie pour vous automatiquement.

Les sections suivantes fournissent quelques détails supplémentaires.

Autoriser la colle : BatchCreatePartition dans la politique IAM

Vérifiez les politiques IAM associées au rôle que vous utilisez pour exécuter `MSCK REPAIR TABLE`. Lorsque vous [utilisez le AWS Glue Data Catalog with Athena](#), la politique IAM doit autoriser l'action. `glue:BatchCreatePartition` Pour un exemple de politique IAM qui autorise l'action `glue:BatchCreatePartition`, voir [AWS politique gérée : AmazonAthenaFullAccess](#).

Modifier ou redéfinir le chemin Amazon S3

Si une ou plusieurs clés d'objet du chemin Amazon S3 sont en casse mixte plutôt qu'en minuscules, `MSCK REPAIR TABLE` peut ne pas ajouter les partitions au AWS Glue Data Catalog. Par exemple, si votre chemin Amazon S3 inclut le nom de la clé de l'objet `userId`, les partitions suivantes peuvent ne pas être ajoutées au AWS Glue Data Catalog :

```
s3://DOC-EXAMPLE-BUCKET/path/userId=1/

s3://DOC-EXAMPLE-BUCKET/path/userId=2/

s3://DOC-EXAMPLE-BUCKET/path/userId=3/
```

Pour résoudre ce problème, procédez de l'une des manières suivantes :

- Utilisez des minuscules au lieu d'une casse mixte lorsque vous créez vos clés d'objet Amazon S3 :

```
s3://DOC-EXAMPLE-BUCKET/path/userid=1/

s3://DOC-EXAMPLE-BUCKET/path/userid=2/
```

```
s3://DOC-EXAMPLE-BUCKET/path/userid=3/
```

- Utilisez ALTER TABLE ADD PARTITION pour redéfinir l'emplacement, comme dans l'exemple suivant :

```
ALTER TABLE table_name ADD [IF NOT EXISTS]
PARTITION (userId=1)
LOCATION 's3://DOC-EXAMPLE-BUCKET/path/userId=1/'
PARTITION (userId=2)
LOCATION 's3://DOC-EXAMPLE-BUCKET/path/userId=2/'
PARTITION (userId=3)
LOCATION 's3://DOC-EXAMPLE-BUCKET/path/userId=3/'
```

Notez que même si les noms de clé d'objet Amazon S3 peuvent être utilisés en majuscules, les noms de compartiment Amazon S3 eux-mêmes doivent toujours être en minuscules. Pour plus d'informations, consultez Directives de dénomination des clés d'objet et Règles de dénomination des compartiments dans le Guide de l'utilisateur Amazon S3.

SHOW COLUMNS

Affiche uniquement les noms de colonnes d'une seule table ou vue spécifiée. Pour obtenir des informations plus détaillées, interrogez AWS Glue Data Catalog plutôt le. Pour des informations et des exemples, consultez les sections suivantes dans la rubrique Interrogation du AWS Glue Data Catalog :

- Pour afficher des métadonnées de colonne telles que le type de données, consultez Liste ou recherche de colonnes pour une table ou une vue spécifiée.
- Pour afficher toutes les colonnes de toutes les tables d'une base de données spécifique dans `AwsDataCatalog`, consultez Liste ou recherche de colonnes pour une table ou une vue spécifiée.
- Pour afficher toutes les colonnes de toutes les tables de toutes les bases de données dans `AwsDataCatalog`, consultez Liste de toutes les colonnes pour toutes les tables.
- Pour afficher les colonnes communes à des tables spécifiques d'une base de données, consultez Affichage des colonnes communes à des tables spécifiques.

Résumé

```
SHOW COLUMNS {FROM|IN} database_name.table_name
```

```
SHOW COLUMNS {FROM|IN} table_name [{FROM|IN} database_name]
```

Les mots-clés FROM et IN peuvent être utilisés de façon interchangeable. Si *nom_de_table* ou *nom_de_base_de_données* comporte des caractères spéciaux comme des tirets, entourez le nom de guillemets (par exemple, `` `my-database`.`my-table` ``). N'entourez pas le *nom_de_table* ou le *nom_de_base* de guillemets simples ou doubles. Actuellement, l'utilisation de LIKE et d'expressions de correspondance de motifs n'est pas prise en charge.

Exemples

Les exemples équivalents suivants montrent les colonnes de la table orders dans la base de données customers. Les deux premiers exemples supposent que customers est la base de données actuelle.

```
SHOW COLUMNS FROM orders
```

```
SHOW COLUMNS IN orders
```

```
SHOW COLUMNS FROM customers.orders
```

```
SHOW COLUMNS IN customers.orders
```

```
SHOW COLUMNS FROM orders FROM customers
```

```
SHOW COLUMNS IN orders IN customers
```

SHOW CREATE TABLE

Analyse une table existante nommée table_name pour générer la requête qui l'a créée.

Résumé

```
SHOW CREATE TABLE [db_name.]table_name
```

Paramètres

TABLE [db_name.]table_name

Le paramètre db_name est facultatif. S'il est omis, le contexte utilise la base de données actuelle par défaut.

> **Note**
>
> Le nom de table est obligatoire.

Exemples

```
SHOW CREATE TABLE orderclickstoday;
```

```
SHOW CREATE TABLE `salesdata.orderclickstoday`;
```

Résolution des problèmes

Si vous utilisez l'opération AWS Glue CreateTableAPI ou le AWS CloudFormation `AWS::Glue::Table`modèle pour créer une table à utiliser dans Athena sans spécifier la `TableType` propriété, puis si vous exécutez une requête DDL du type `SHOW CREATE TABLE` ou`MSCK REPAIR TABLE`, vous pouvez recevoir le message d'erreur FAILED : NullPointerException Name is null.

Pour résoudre l'erreur, spécifiez une valeur pour l'TableInput`TableType`attribut dans le cadre de l'appel ou du AWS CloudFormation modèle d' AWS Glue `CreateTable`API. Parmi les valeurs possibles pour `TableType` figurent `EXTERNAL_TABLE` ou `VIRTUAL_VIEW`.

Cette exigence s'applique uniquement lorsque vous créez une table à l'aide de l'opération AWS Glue `CreateTable` API ou du `AWS::Glue::Table` modèle. Si vous créez une table pour Athena en utilisant à l'aide d'une instruction DDL ou d'un crawler AWS Glue , la propriété `TableType` est définie pour vous automatiquement.

SHOW CREATE VIEW

Affiche l'instruction SQL qui crée la vue spécifiée.

Résumé

```
SHOW CREATE VIEW view_name
```

Exemples

```
SHOW CREATE VIEW orders_by_date
```

Voir aussi CREATE VIEW et DROP VIEW.

SHOW DATABASES

Répertorie toutes les bases de données définies dans le metastore. Vous pouvez utiliser `DATABASES` ou `SCHEMAS`. Ils ont la même signification.

L'équivalent programmatique de `SHOW DATABASES` est l'action de l'ListDatabasesAPI Athena. La méthode équivalente dans AWS SDK for Python (Boto3) est list_databases.

Résumé

```
SHOW {DATABASES | SCHEMAS} [LIKE 'regular_expression']
```

Paramètres

[LIKE '***regular_expression***']

Filtre la liste des bases de données sur celles qui correspondent à l'expression régulière *`regular_expression`* que vous spécifiez. Pour la correspondance de caractères génériques, vous pouvez utiliser la combinaison `.*`, qui correspond à n'importe quel caractère indépendamment de la fréquence d'utilisation.

Exemples

```
SHOW SCHEMAS;
```

```
SHOW DATABASES LIKE '.*analytics';
```

SHOW PARTITIONS

Répertorie toutes les partitions d'une table Athena dans un ordre non trié.

Résumé

```
SHOW PARTITIONS table_name
```

- Pour afficher les partitions dans un tableau et les répertorier selon un ordre spécifique, consultez la section Liste des partitions d'une table spécifique de la page Interrogation du AWS Glue Data Catalog.
- Pour afficher le contenu d'une partition, consultez la section Exécution des requêtes de données de la page Partitionnement de données dans Athena.
- `SHOW PARTITIONS`ne répertorie pas les partitions projetées par Athena mais non enregistrées dans le AWS Glue catalogue. Pour plus d'informations sur la projection de partition, consultez Projection de partition avec Amazon Athena.
- `SHOW PARTITIONS` répertorie les partitions dans les métadonnées, et non les partitions dans le système de fichiers réel. Pour mettre à jour les métadonnées après avoir supprimé manuellement des partitions dans Simple Storage Service (Amazon S3), exécutez ALTER TABLE DROP PARTITION.

Exemples

L'exemple de requête suivant montre les partitions de la table `flight_delays_csv` , qui présente les données de la table des vols du ministère américain des transports. Pour plus d'informations sur l'exemple de table `flight_delays_csv`, voir LazySimpleSerDe pour les fichiers CSV, TSV et délimités de manière personnalisée. La table est partitionnée par année.

```
SHOW PARTITIONS flight_delays_csv
```

Résultats

```
year=2007
year=2015
year=1999
year=1993
year=1991
year=2003
year=1996
year=2014
year=2004
year=2011
```

```
...
```

L'exemple de requête suivant montre les partitions de la table `impressions`, qui contient des exemples de données de navigation web. Pour plus d'informations sur l'exemple de table `impressions`, voir Partitionnement de données dans Athena. La table est partitionnée par la colonne `dt` (date et heure).

```
SHOW PARTITIONS impressions
```

Résultats

```
dt=2009-04-12-16-00
dt=2009-04-13-18-15
dt=2009-04-14-00-20
dt=2009-04-12-13-00
dt=2009-04-13-02-15
dt=2009-04-14-12-05
dt=2009-04-14-06-15
dt=2009-04-12-21-15
dt=2009-04-13-22-15
...
```

Liste des partitions dans un ordre trié

Pour ordonner les partitions dans la liste des résultats, utilisez la syntaxe `SELECT` suivante au lieu de `SHOW PARTITIONS`.

```
SELECT * FROM database_name."table_name$partitions" ORDER BY column_name
```

La requête suivante montre la liste des partitions pour l'exemple `flight_delays_csv`, mais dans un ordre trié.

```
SELECT * FROM "flight_delays_csv$partitions" ORDER BY year
```

Résultats

```
year
1987
1988
1989
```

```
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
...
```

Pour de plus amples informations, consultez la section Liste des partitions d'une table spécifique de la page Interrogation du AWS Glue Data Catalog.

SHOW TABLES

Affiche toutes les tables de base et vues d'une base de données.

Résumé

```
SHOW TABLES [IN database_name] ['regular_expression']
```

Paramètres

[IN nom_basededonnées]

Spécifie la base de données `database_name` à partir de laquelle les tables sont affichées. Si la valeur est omise, la base de données du contexte en cours est utilisée par défaut.

> **Note**
>
> `SHOW TABLES` peut échouer si `database_name` utilise un caractère non pris en charge comme un trait d'union. Pour contourner ce problème, essayez d'entourer le nom de la base de données de guillemets.

['expression_régulière']

Filtre la liste des tables sur celles qui correspondent à l'expression régulière `regular_expression` que vous spécifiez. Pour indiquer n'importe quel caractère dans les

tables `AWSDataCatalog` , vous pouvez utiliser l'expression générique `*` ou `.*`. Pour les bases de données Apache Hive, utilisez l'expression générique `.*`. Pour indiquer un choix entre plusieurs caractères, utilisez le caractère `|`.

Exemples

Example – Afficher toutes les tables de la base de données **`sampledb`**

```
SHOW TABLES IN sampledb
```

`Results`

```
alb_logs
cloudfront_logs
elb_logs
flights_2016
flights_parquet
view_2016_flights_dfw
```

Example – Afficher les noms de toutes les tables de **`sampledb`** qui incluent le mot « vols »

```
SHOW TABLES IN sampledb '*flights*'
```

`Results`

```
flights_2016
flights_parquet
view_2016_flights_dfw
```

Example – Afficher les noms de toutes les tables de **`sampledb`** qui se terminent par le mot « journaux »

```
SHOW TABLES IN sampledb '*logs'
```

`Results`

```
alb_logs
cloudfront_logs
elb_logs
```

SHOW TBLPROPERTIES

Répertorie les propriétés de table pour la table nommée.

Résumé

```
SHOW TBLPROPERTIES table_name [('property_name')]
```

Paramètres

[('property_name')]

Si ce paramètre est inclus, seule la valeur de la propriété nommée `property_name` est répertoriée.

Exemples

```
SHOW TBLPROPERTIES orders;
```

```
SHOW TBLPROPERTIES orders('comment');
```

SHOW VIEWS

Répertorie les vues dans la base de données spécifiée, ou dans la base de données actuelle, si vous ne spécifiez pas le nom de base de données. Utilisez la clause `LIKE` en option avec une expression régulière pour restreindre la liste des noms de vue.

Athena renvoie une liste de valeurs de type `STRING` dans laquelle chaque valeur est un nom de vue.

Résumé

```
SHOW VIEWS [IN database_name] [LIKE 'regular_expression']
```

Paramètres

[IN nom_basededonnées]

Spécifie la `database_name` à partir de laquelle les vues sont affichées. Si la valeur est omise, la base de données du contexte en cours est utilisée par défaut.

[LIKE 'regular_expression']

Filtre la liste des vues sur celles qui correspondent à l'expression régulière `regular_expression` que vous spécifiez. Seul le caractère générique `*`, qui désigne n'importe quel caractère, ou `|`, qui indique un choix entre caractères, peuvent être utilisés.

Exemples

```
SHOW VIEWS;
```

```
SHOW VIEWS IN marketing_analytics LIKE 'orders*'
```

Consultez aussi SHOW COLUMNS, SHOW CREATE VIEW, DESCRIBE VIEW et DROP VIEW.

Considérations et limitations relatives aux requêtes SQL dans Amazon Athena

Lorsque vous exécutez des requêtes dans Athena, gardez à l'esprit les considérations et limitations suivantes :

- Procédures stockées – Les procédures stockées ne sont pas prises en charge.
- Nombre maximum de partitions – Le nombre maximum de partitions que vous pouvez créer avec des instructions `CREATE TABLE AS SELECT` (CTAS) est de 100. Pour obtenir des informations, consultez CREATE TABLE AS. Pour obtenir une solution de contournement, veuillez consulter Utilisation de CTAS et de INSERT INTO pour contourner la limite de 100 partitions.
- Instructions non prises en charge – Les instructions suivantes ne sont pas prises en charge :
  - `CREATE TABLE LIKE` n'est pas pris en charge.
  - `DESCRIBE INPUT` et `DESCRIBE OUTPUT` ne sont pas pris en charge.
  - L'instruction `MERGE` n'est prise en charge que pour les formats de table transactionnels. Pour plus d'informations, consultez MERGE INTO.
  - Les instructions `UPDATE` ne sont pas prises en charge.
- Connecteurs Trino et Presto : les connecteurs Trino et Presto ne sont pas pris en charge. Utilisation d'une requête fédérée d'Amazon Athena pour vous connecter aux sources de données. Pour plus d'informations, consultez Utilisation de la requête fédérée d'Amazon Athena.
- Délais d'expiration sur des tables avec plusieurs partitions – Athena peut expirer lors de l'interrogation d'une table qui contient plusieurs milliers de partitions. Cela peut se produire lorsque

la table comporte de nombreuses partitions qui ne sont pas de type `string`. Lorsque vous utilisez le type `string`, Athena réduit les partitions au niveau du métastore. Toutefois, lorsque vous utilisez d'autres types de données, Athena réduit les partitions côté serveur. Plus vous avez de partitions, plus ce processus est long et plus vos requêtes sont susceptibles d'expirer. Pour résoudre ce problème, définissez votre type de partition sur `string` de sorte qu'Athena réduise les partitions au niveau du métastore. Cela réduit les frais généraux et empêche les requêtes d'expirer.

- Support de S3 Glacier : pour plus d'informations sur l'interrogation d'objets Amazon S3 Glacier restaurés, consultez Interrogation d'objets Amazon S3 Glacier restaurés.
- Fichiers traités comme masqués – Athena traite les fichiers sources qui commencent par un trait de soulignement (_) ou un point (`.`) comme étant masqués. Pour contourner cette limitation, renommez les fichiers.
- Limitation de taille de ligne ou de colonne – La taille d'une ligne ou de ses colonnes ne peut pas dépasser 32 mégaoctets. Cette limite peut être dépassée lorsque, par exemple, une ligne d'un fichier CSV ou JSON contient une seule colonne de 300 mégaoctets. Le dépassement de cette limite peut également produire le message d'erreurLine too long in text file (Ligne trop longue dans le fichier texte). Pour contourner cette limitation, assurez-vous que la somme des données des colonnes de chaque ligne est inférieure à 32 Mo.
- Maximum de la clause LIMIT : le nombre maximum de lignes pouvant être spécifié pour la clause `LIMIT` est

  9223372036854775807. Lors de l'utilisation de `ORDER BY`, le nombre maximum de lignes prises en charge pour la clause LIMIT est 2 147 483 647. Le dépassement de cette limite renvoie le message d'erreur NOT_SUPPORTED: ORDER BY LIMIT > 2147483647 is not supported (NOT_SUPPORTED : ORDER BY LIMIT > 2 147 483 647 n'est pas pris en charge).
- information_schema — Les requêtes `information_schema` sont plus performantes si vous disposez d'une quantité faible à modérée de métadonnées. AWS Glue Si vous disposez d'un grand nombre de métadonnées, des erreurs peuvent se produire. Pour plus d'informations sur l'interrogation de AWS Glue métadonnées dans la `information_schema` base de données, consultezInterrogation du AWS Glue Data Catalog.
- Initialisations de tableaux – En raison d'une limitation de Java, il n'est pas possible d'initialiser dans Athena un tableau comportant plus de 254 arguments.
- Colonnes de métadonnées masquées : les colonnes de métadonnées masquées Hive ou Iceberg `$bucket`, `$file_modified_time`, `$file_size` et `$partition` ne sont pas prises en charge pour les vues. Pour plus d'informations sur l'utilisation de la colonne de métadonnées `$path` dans

Athena, consultez Obtention des emplacements de fichiers pour les données source dans Simple Storage Service (Amazon S3) .

Pour plus d'informations sur la longueur maximale de la chaîne de requête, les quotas des délais d'expiration des requêtes et les quotas du nombre actif de requêtes DML, consultez Service Quotas.

Résolution des problèmes dans Athena

L'équipe d'Athena a rassemblé les informations suivantes de résolution de problèmes à partir des problèmes des clients. Bien qu'elles ne soient pas exhaustives, elles comprennent des conseils concernant certains problèmes courants de performances, de délais d'attente et de manque de mémoire.

Rubriques

CREATE TABLE AS SELECT (CTAS)

Il y a duplication des données en cas d'instructions CTAS simultanées

Athena ne maintient pas la validation simultanée pour CTAS. Assurez-vous qu'il n'y a pas d'instruction CTAS dupliquée pour le même emplacement au même moment. Même si une instruction CTAS ou INSERT INTO échoue, les données orphelines peuvent être laissées dans l'emplacement de données spécifié dans l'instruction.

HIVE_TOO_MANY_OPEN_PARTITIONS

Lorsque vous utilisez une instruction CTAS pour créer une table comportant plus de 100 partitions, l'erreur HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets (Dépassement de la limite de 100 rédacteurs ouverts pour les partitions/compartiments) peut s'afficher. Pour contourner ces limitations, vous pouvez utiliser une instruction CTAS et une série d'instructions `INSERT INTO` qui créent ou insèrent jusqu'à 100 partitions chacune. Pour plus d'informations, consultez Utilisation de CTAS et de INSERT INTO pour contourner la limite de 100 partitions.

Problèmes de fichiers de données

Athena ne peut pas lire les fichiers cachés

Athena traite les fichiers sources qui commencent par un trait de soulignement (_) ou un point (.) comme étant cachés. Pour contourner cette limitation, renommez les fichiers.

Athena lit les fichiers que j'ai exclus du crawler AWS Glue

Athena ne reconnaît pas les modèles d'exclusion que vous spécifiez à un AWS Glue robot d'exploration. Par exemple, si vous disposez d'un compartiment Simple Storage Service (Amazon S3) contenant à la fois des fichiers `.csv` et `.json` et que vous excluez les fichiers `.json` du Crawler, Athena interroge les deux groupes de fichiers. Pour éviter cela, placez les fichiers que vous voulez exclure dans un autre emplacement.

HIVE_BAD_DATA : erreur lors de l'analyse de la valeur du champ

Cette erreur peut se produire dans les cas suivants :

- Le type de données défini dans la table ne correspond pas aux données sources, ou un seul champ contient différents types de données. Pour les résolutions suggérées, consultez la rubrique

Ma requête Amazon Athena échoue avec l'erreur « HIVE_BAD_DATA: Error parsing field value for field x: For input string: "12312845691" » (HIVE_BAD_DATA : erreur d'analyse de la valeur du champ x : pour la chaîne d'entrée : "12312845691") dans le Centre de connaissances AWS .

- Des valeurs nulles sont présentes dans un champ de type entier. Une solution consiste à créer la colonne avec les valeurs nulles en tant que `string` et utiliser ensuite `CAST` pour convertir le champ dans une requête en fournissant une valeur par défaut de `0` pour les valeurs nulles. Pour plus d'informations, consultez la rubrique Lorsque j'interroge des données CSV dans Athena, je reçois l'erreur « HIVE_BAD_DATA: Error parsing field value " for field x: For input string: " » (HIVE_BAD_DATA : erreur d'analyse de la valeur du champ " pour le champ x : pour la chaîne d'entrée : ") dans le Centre de connaissances AWS .

HIVE_CANNOT_OPEN_SPLIT : Erreur lors de l'ouverture de Hive Split s3://DOC-EXAMPLE-BUCKET

Cette erreur peut se produire lorsque vous interrogez un préfixe de compartiment Simple Storage Service (Amazon S3) qui contient un grand nombre d'objets. Pour plus d'informations, consultez Comment résoudre l'erreur « HIVE_CANNOT_OPEN_SPLIT : Error opening Hive split s3://DOC-EXAMPLE-BUCKET/ : Slow down » dans Athena ? dans le AWS Knowledge Center.

HIVE_CURSOR_ERROR : com.amazonaws.services.s3.model.AmazonS3Exception : la clé spécifiée n'existe pas

Cette erreur se produit généralement lorsqu'un fichier est supprimé alors qu'une requête est en cours d'exécution. Vous pouvez soit réexécuter la requête, soit vérifier votre flux de travail pour voir si une autre tâche ou un autre processus modifie les fichiers pendant l'exécution de la requête.

HIVE_CURSOR_ERROR : fin inattendue du flux d'entrée

Ce message indique que le fichier est soit corrompu, soit vide. Vérifiez l'intégrité du fichier et exécutez à nouveau la requête.

HIVE_FILESYSTEM_ERROR : taille du fichier ***1234567*** incorrecte pour le fichier

Ce message peut survenir lorsqu'un fichier a été modifié entre la planification de la requête et son exécution. Cela se produit généralement lorsqu'un fichier sur Simple Storage Service (Amazon S3) est remplacé sur place (par exemple, un PUT est exécuté sur une clé où un objet existe déjà). Athena ne prend pas en charge la suppression ou le remplacement du contenu d'un fichier lorsqu'une requête est en cours d'exécution. Pour éviter cette erreur, planifiez des tâches qui écrasent ou

suppriment des fichiers à des moments où les requêtes ne sont pas exécutées, ou qui n'écrivent des données que dans de nouveaux fichiers ou partitions.

HIVE_UNKNOWN_ERROR : impossible de créer le format d'entrée

Cette erreur peut être le résultat de problèmes tels que :

- Le AWS Glue robot n'a pas pu classer le format des données
- Certaines propriétés de définition de AWS Glue table sont vides
- Athena ne prend pas en charge le format de données des fichiers de Simple Storage Service (Amazon S3)

Pour plus d'informations, consultez la rubrique Comment résoudre l'erreur « impossible de créer le format d'entrée » dans Athena ? dans le Centre de connaissances AWS ou regardez la vidéo du Centre de connaissances.

L'emplacement S3 fourni pour enregistrer les résultats de votre requête n'est pas valide.

Assurez-vous que vous avez spécifié un emplacement S3 valide pour les résultats de votre requête. Pour plus d'informations, consultez Spécification d'un emplacement de résultats de requête dans la rubrique Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie.

Tables Linux Foundation Delta Lake

Schéma de table Delta Lake non synchronisé

Lorsque vous interrogez une table Delta Lake dont le schéma est obsolète, le message d'erreur suivant peut s'afficher : AWS Glue

```
INVALID_GLUE_SCHEMA: Delta Lake table schema in Glue does not match the most recent
 schema of the
Delta Lake transaction log. Please ensure that you have the correct schema defined in
 Glue.
```

Le schéma peut devenir obsolète s'il est modifié AWS Glue après avoir été ajouté à Athena. Pour mettre à jour le schéma, effectuez l'une des étapes suivantes :

- Entrez AWS Glue, lancez le AWS Glue crawler.

- Dans Athéna, supprimez la table et créez-la à nouveau.
- Ajoutez les colonnes manquantes manuellement, soit en utilisant l'instruction ALTER TABLE ADD COLUMNS dans Athena, soit en modifiant le schéma de table dans AWS Glue.

Requêtes fédérées

Délai d'attente pendant l'appel ListTableMetadata

Un appel à l'ListTableMetadataAPI peut expirer si la source de données contient de nombreuses tables, si la source de données est lente ou si le réseau est lent. Pour résoudre ce problème, essayez les étapes suivantes.

- Vérifier le nombre de tables : si vous avez plus de 1 000 tables, essayez de réduire le nombre de tables. Pour obtenir la réponse `ListTableMetadata` la plus rapide, nous vous recommandons d'avoir moins de 1 000 tables par catalogue.
- Vérifier la configuration Lambda – Il est essentiel de surveiller le comportement de la fonction Lambda. Lorsque vous utilisez des catalogues fédérés, veillez à examiner les journaux d'exécution de la fonction Lambda. Sur la base des résultats, ajustez les valeurs de mémoire et de délai d'expiration en conséquence. Pour identifier tout problème potentiel lié aux délais d'expiration, parcourez à nouveau votre configuration Lambda. Pour plus d'informations, consultez Configuration du délai d'expiration de la fonction (console) dans le Guide du développeur AWS Lambda .
- Vérifier les journaux des sources de données fédérées – Examinez les journaux et les messages d'erreur de la source de données fédérée pour voir s'il existe des problèmes ou des erreurs. Les journaux peuvent fournir des informations précieuses sur la cause du délai d'expiration.
- Utiliser **`StartQueryExecution`** pour récupérer les métadonnées : si vous avez plus de 1 000 tables, la récupération des métadonnées à l'aide de votre connecteur fédéré peut prendre plus de temps que prévu. Étant donné que la nature asynchrone de StartQueryExecutiongarantit qu'Athena exécute la requête de la manière la plus optimale, envisagez de l'utiliser `StartQueryExecution` comme alternative à. `ListTableMetadata` Les AWS CLI exemples suivants montrent `StartQueryExecution` comment utiliser au lieu d'`ListTableMetadata`obtenir toutes les métadonnées des tables de votre catalogue de données.

  Commencez par exécuter une requête qui récupère toutes les tables, comme dans l'exemple suivant.

```
aws athena start-query-execution --region us-east-1 \
--query-string "SELECT table_name FROM information_schema.tables LIMIT 50" \
--work-group "your-work-group-name"
```

Ensuite, récupérez les métadonnées d'une table individuelle, comme dans l'exemple suivant.

```
aws athena start-query-execution --region us-east-1 \
--query-string "SELECT * FROM information_schema.columns \
WHERE table_name = 'your-table-name' AND \
table_catalog = 'your-catalog-name'" \
--work-group "your-work-group-name"
```

Le temps nécessaire pour obtenir les résultats dépend du nombre de tables de votre catalogue.

Pour plus d'informations sur la résolution des problèmes liés aux requêtes fédérées, consultez Common_Problems dans la aws-athena-query-federation section awslabs/ de GitHub, ou consultez la documentation des connecteurs de source de données Athena individuels.

Erreurs liées à JSON

Erreurs de données NULL ou incorrectes lors de la tentative de lecture de données JSON

Les erreurs de données NULL ou incorrectes lorsque vous essayez de lire des données JSON peuvent avoir plusieurs causes. Pour identifier les lignes à l'origine des erreurs lorsque vous utilisez OpenX SerDe, définissez sur`ignore.malformed.json`. `true` Les registres mal formés seront renvoyés comme NULL. Pour plus d'informations, consultez la rubrique Je reçois des erreurs lorsque j'essaie de lire des données JSON dans Amazon Athena dans le Centre de connaissances AWS ou regardez la vidéo du Centre de connaissances.

HIVE_BAD_DATA : erreur d'analyse de la valeur du champ 0 : java.lang.String ne peut pas être converti en org.openx.data.jsonserde.json.JSONObject

Le SerDe OpenX JSON SerDe renvoie cette erreur lorsqu'il ne parvient pas à analyser une colonne dans une requête Athena. Cela peut se produire si vous définissez une colonne comme un `map` ou un `struct`, mais que les données sous-jacentes sont en fait un `string`, un `int` ou un autre type primitif.

HIVE_CURSOR_ERROR : la ligne n'est pas un objet JSON valide - JSONException : clé dupliquée

Cette erreur se produit lorsque vous utilisez Athena pour interroger AWS Config des ressources comportant plusieurs balises portant le même nom dans différents cas. La solution consiste à exécuter `CREATE TABLE` en utilisant `WITH SERDEPROPERTIES 'case.insensitive'='false'` et à mapper les noms. Pour plus d'informations sur `case.insensitive` et le mappage, voir SerDe bibliothèques JSON. Pour plus d'informations, voir Comment résoudre « HIVE_CURSOR_ERROR : Row is not a valid JSON object - JSONException : Duplicate key » lors de la lecture de fichiers depuis Athena ? AWS Config dans le AWS Knowledge Center.

Messages HIVE_CURSOR_ERROR avec JSON formaté à des fins d'impression (pretty-printed)

Les bibliothèques Hive JSON SerDe et OpenX JSON SerDe nécessitent que chaque document JSON soit sur une seule ligne de texte, sans caractères de fin de ligne pour séparer les champs de l'enregistrement. Si le texte JSON est dans un joli format d'impression, vous pouvez recevoir un message d'erreur tel que HIVE_CURSOR_ERROR : Row is not a valid JSON Object ou HIVE_CURSOR_ERROR : : Unexpected JsonParseException end-of-input : expected close marker for OBJECT lorsque vous essayez d'interroger la table après l'avoir créée. Pour plus d'informations, consultez les fichiers de données JSON dans la SerDe documentation OpenX sur. GitHub

Plusieurs registres JSON renvoient un SELECT COUNT de 1

Si vous utilisez le SerDe OpenX JSON SerDe, assurez-vous que les registres sont séparés par un caractère de saut de ligne. Pour plus d'informations, consultez La requête SELECT COUNT dans Amazon Athena ne renvoie qu'un seul enregistrement, même si le fichier JSON d'entrée contient plusieurs enregistrements dans le AWS Knowledge Center.

Impossible d'interroger une table créée par un AWS Glue robot d'exploration qui utilise un classificateur JSON personnalisé

Le moteur Athena ne prend pas en charge les classificateurs JSON personnalisés. Pour contourner ce problème, créez une nouvelle table sans le classificateur personnalisé. Pour transformer le JSON, vous pouvez utiliser CTAS ou créer une vue. Par exemple, si vous travaillez avec des tableaux, vous pouvez utiliser l'option UNNEST pour aplatir le JSON. Une autre option consiste à utiliser une tâche AWS Glue ETL qui prend en charge le classificateur personnalisé, à convertir les données en parquet dans Amazon S3, puis à les interroger dans Athena.

MSCK REPAIR TABLE

Pour plus d'informations sur les problèmes liés à MSCK REPARATION TABLE, consultez les sections Considérations et restrictions et Résolution des problèmes de la page MSCK REPAIR TABLE.

Problèmes de sortie

Impossible de vérifier/créer un compartiment de sortie

Cette erreur peut se produire si l'emplacement spécifié pour le résultat de la requête n'existe pas ou si les autorisations appropriées ne sont pas présentes. Pour plus d'informations, consultez Comment résoudre l'erreur « Impossible de vérifier/créer un compartiment de sortie » dans Amazon Athena ? dans le AWS Knowledge Center.

Le résultat TIMESTAMP est vide

Athena nécessite le format TIMESTAMP de Java. Pour plus d'informations, consultez la rubrique Lorsque j'interroge une table dans Amazon Athena, le résultat TIMESTAMP est vide dans le Centre de connaissances AWS .

Stockage des résultats de la requête Athena dans un format autre que CSV

Par défaut, Athena n'affiche que des fichiers au format CSV. Pour afficher les résultats d'une requête `SELECT` dans un autre format, vous pouvez utiliser l'instruction `UNLOAD`. Pour plus d'informations, consultez UNLOAD. Vous pouvez également utiliser une requête CTAS qui utilise la propriété de table `format` pour configurer le format de sortie. Contrairement à `UNLOAD`, la technique CTAS nécessite la création d'une table. Pour plus d'informations, voir Comment puis-je stocker une sortie de requête Athena dans un format autre que CSV, tel qu'un format compressé ? dans le AWS Knowledge Center.

L'emplacement S3 fourni pour enregistrer les résultats de votre requête n'est pas valide

Il est possible que vous receviez ce message d'erreur si l'emplacement de votre compartiment de sortie ne se trouve pas dans la même région que celle dans laquelle vous exécutez votre requête. Pour éviter cela, spécifiez l'emplacement des résultats de la requête dans la région dans laquelle vous exécutez la requête. Pour les étapes, consultez Spécification d'un emplacement de résultats de requête.

Problèmes liés à Parquet

org.apache.parquet.io. GroupColumnL'IO ne peut pas être converti vers org.apache.parquet.io. PrimitiveColumnIO

Cette erreur est causée par une inadéquation du schéma Parquet. Une colonne qui a un type non primitif (par exemple, `array`) a été déclarée comme un type primitif (par exemple, `string`) dans AWS Glue. Pour résoudre ce problème, vérifiez le schéma de données dans les fichiers et comparez-le au schéma déclaré dans AWS Glue.

Problèmes de statistiques liés à Parquet

Lorsque vous lisez des données Parquet, vous pouvez recevoir des messages d'erreur tels que les suivants :

```
HIVE_CANNOT_OPEN_SPLIT: Index x out of bounds for length y
HIVE_CURSOR_ERROR: Failed to read x bytes
HIVE_CURSOR_ERROR: FailureException at Malformed input: offset=x
HIVE_CURSOR_ERROR: FailureException at java.io.IOException:
can not read class org.apache.parquet.format.PageHeader: Socket is closed by peer.
```

Pour contourner ce problème, utilisez l'ALTER TABLE SET TBLPROPERTIESinstruction CREATE TABLE or pour définir la SerDe `parquet.ignore.statistics` propriété Parquet sur`true`, comme dans les exemples suivants.

Exemple CREATE TABLE

```
...
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES ('parquet.ignore.statistics'='true')
STORED AS PARQUET
...
```

Exemple ALTER TABLE

```
ALTER TABLE ... SET TBLPROPERTIES ('parquet.ignore.statistics'='true')
```

Pour plus d'informations sur le Parquet Hive SerDe, consultezParquet SerDe.

Problèmes de partitionnement

MSCK REPARATION TABLE ne supprime pas les partitions périmées

Si vous supprimez manuellement une partition dans Simple Storage Service (Amazon S3), puis exécutez MSCK REPAIR TABLE, il est possible que vous receviez le message d'erreur Partitions missing from filesystem (Partitions manquantes dans le système de fichiers). Cela se produit parce que MSCK REPAIR TABLE ne supprime pas les partitions périmées des métadonnées de la table. Utilisez ALTER TABLE DROP PARTITION pour supprimer manuellement les partitions périmées. Pour plus d'informations, consultez la section « Résolution des problèmes » de la rubrique MSCK REPAIR TABLE.

Échec de MSCK REPAIR TABLE

Lorsqu'un grand nombre de partitions (par exemple, plus de 100 000) sont associées à une table particulière, `MSCK REPAIR TABLE` peut échouer en raison de limitations de mémoire. Pour contourner cette limite, utilisez ALTER TABLE ADD PARTITION à la place.

MSCK REPAIR TABLE détecte les partitions mais ne les ajoute pas AWS Glue

Ce problème peut se produire si un chemin d'accès Simple Storage Service (Amazon S3) est en majuscules au lieu de minuscules ou si une politique IAM n'autorise pas l'action `glue:BatchCreatePartition`. Pour plus d'informations, consultez MSCK REPAIR TABLE détecte les partitions dans Athena mais ne les ajoute pas AWS Glue Data Catalog dans le Knowledge Center. AWS

Les plages de projection des partitions avec le format de date jj-MM-aaaa-HH-mm-ss ou aaaa-MM-jj ne fonctionnent pas

Pour fonctionner correctement, le format de date doit être défini sur `yyyy-MM-dd HH:00:00`. Pour plus d'informations, consultez l'article Dépassement de la capacité de la pile : la projection de partition Athena ne fonctionne pas comme prévu.

PARTITION BY ne prend pas en charge le type BIGINT

Convertir le type de données en `string` et réessayer.

Pas de partitions significatives disponibles

Ce message d'erreur signifie généralement que les paramètres de la partition ont été corrompus. Pour résoudre ce problème, abandonnez la table et créez une table avec de nouvelles partitions.

La projection de partition ne fonctionne pas en conjonction avec les partitions de plage

Vérifiez que l'unité de plage de temps projection.*<columnName>*.interval.unit correspond au délimiteur des partitions. Par exemple, si les partitions sont délimitées par des jours, l'unité de plage des heures ne fonctionnera pas.

Erreur de projection de partition lorsque la plage est spécifiée par un tiret

La spécification de la propriété de table `range` avec un tiret au lieu d'une virgule produit une erreur du type INVALID_TABLE_PROPERTY : Pour la chaîne de saisie : « *number*-*number* ». Veillez à ce que les valeurs de la plage soient séparées par une virgule, et non par un tiret. Pour plus d'informations, consultez Type d'entier.

HIVE_UNKNOWN_ERROR : impossible de créer le format d'entrée

Une ou plusieurs des partitions Glue sont déclarées dans un format différent, car chaque partition Glue a son propre format d'entrée spécifique de manière indépendante. Vérifiez comment vos partitions sont définies dans AWS Glue.

HIVE_PARTITION_SCHEMA_MISMATCH

Si le schéma d'une partition diffère du schéma de la table, la requête peut échouer avec le message d'erreur HIVE_PARTITION_SCHEMA_MISMATCH. Pour plus d'informations, consultez Synchronisation du schéma de la partition pour éviter « HIVE_PARTITION_SCHEMA_MISMATCH ».

SemanticException la table n'est pas partitionnée mais la spécification de partition existe

Cette erreur peut se produire lorsqu'aucune partition n'a été définie dans l'instruction `CREATE TABLE`. Pour plus d'informations, consultez Comment puis-je résoudre l'erreur « ÉCHEC : la SemanticException table n'est pas partitionnée mais la spécification de partition existe » dans Athena ? dans le AWS Knowledge Center.

Fichiers **`_$folder$`** zéro octet

Si vous exécutez une instruction `ALTER TABLE ADD PARTITION` et spécifiez par erreur une partition déjà existante et un emplacement Simple Storage Service (Amazon S3) incorrect, des fichiers d'emplacement zéro octet du format *`partition_value`*`_$folder$` sont créés dans Simple Storage Service (Amazon S3). Vous devez supprimer ces fichiers manuellement.

Pour éviter que cela ne se produise, utilisez la syntaxe `ADD IF NOT EXISTS` dans votre instruction `ALTER TABLE ADD PARTITION` comme suit :

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITIION […]
```

Aucun registre renvoyé par les données partitionnées

Ce problème peut se produire pour différentes raisons. Pour connaître les causes et les solutions possibles, consultez la section J'ai créé une table dans Amazon Athena avec des partitions définies, mais lorsque j'interroge la table, aucun enregistrement n'est renvoyé dans le AWS Knowledge Center.

Voir aussi HIVE_TOO_MANY_OPEN_PARTITIONS.

Autorisations

Erreur d'accès refusé lors de l'interrogation de Simple Storage Service (Amazon S3)

Cela peut se produire lorsque vous n'avez pas l'autorisation de lire les données dans le compartiment, l'autorisation d'écrire dans le compartiment de résultats ou que le chemin d'accès Simple Storage Service (Amazon S3) contient un point de terminaison de région comme `us-east-1.amazonaws.com`. Pour plus d'informations, consultez la rubrique Lorsque j'exécute une requête Athena, je reçois une erreur « access denied » (Accès refusé) dans le centre de connaissances AWS .

Accès refusé avec le code d'état : erreur 403 lors de l'exécution de requêtes DDL sur des données chiffrées dans Simple Storage Service (Amazon S3)

À quel moment vous pouvez recevoir le message d'erreur Accès refusé (service : Amazon S3 ; code d'état : 403 ; code d'erreur : AccessDenied ; ID de demande :) <request_id>si les conditions suivantes sont remplies :

1. Vous exécutez une requête DDL comme `ALTER TABLE ADD PARTITION` ou `MSCK REPAIR TABLE`.
2. Vous avez un compartiment dont le chiffrement par défaut est configuré pour utiliser `SSE-S3`.
3. Le compartiment a également une politique de compartiment comme la suivante qui force les demandes `PutObject` à spécifier les en-têtes PUT `"s3:x-amz-server-side-encryption": "true"` etd `"s3:x-amz-server-side-encryption": "AES256"`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::<resource-name>/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption": "true"
                }
            }
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::<resource-name>/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "AES256"
                }
            }
        }
    ]
}
```

Dans un cas comme celui-ci, la solution recommandée est de supprimer la politique de compartiment comme celle ci-dessus, étant donné que le chiffrement par défaut du compartiment est déjà présent.

Accès refusé avec le code d'état : 403 lors de l'interrogation d'un compartiment Simple Storage Service (Amazon S3) dans un autre compte

Cette erreur peut se produire lorsque vous essayez d'interroger des journaux rédigés par un autre compte Service AWS et que le second compte est le propriétaire du compartiment mais ne possède pas les objets du compartiment. Pour plus d'informations, consultez l'exception Amazon S3 « accès refusé avec le code de statut : 403 » dans Amazon Athena lorsque j'interroge un bucket dans un autre compte dansAWS le Knowledge Center ou que je regarde la vidéo du Knowledge Center.

Utilisation des informations d'identification du rôle IAM pour la connexion au pilote JDBC Athena

Vous pouvez récupérer les informations d'identification temporaires d'un rôle pour authentifier la Connexion JDBC à Athena. Les informations d'identification temporaires ont une durée de vie maximale de 12 heures. Pour plus d'informations, consultez Comment utiliser les informations d'identification de mon rôle IAM ou passer à un autre rôle IAM lorsque je me connecte à Athena à l'aide du pilote JDBC ? dans le AWS Knowledge Center.

Problèmes de syntaxe des requêtes

ÉCHEC : NullPointerException le nom est nul

Si vous utilisez l'opération AWS Glue CreateTableAPI ou le AWS CloudFormation `AWS::Glue::Table`modèle pour créer une table à utiliser dans Athena sans spécifier la `TableType` propriété, puis si vous exécutez une requête DDL du type `SHOW CREATE TABLE` ou`MSCK REPAIR TABLE`, vous pouvez recevoir le message d'erreur FAILED : NullPointerException Name is null.

Pour résoudre l'erreur, spécifiez une valeur pour l'TableInput`TableType`attribut dans le cadre de l'appel ou du AWS CloudFormation modèle d' AWS Glue `CreateTable`API. Parmi les valeurs possibles pour `TableType` figurent `EXTERNAL_TABLE` ou `VIRTUAL_VIEW`.

Cette exigence s'applique uniquement lorsque vous créez une table à l'aide de l'opération AWS Glue `CreateTable` API ou du `AWS::Glue::Table` modèle. Si vous créez une table pour Athena en utilisant à l'aide d'une instruction DDL ou d'un crawler AWS Glue , la propriété `TableType` est définie pour vous automatiquement.

Fonction non enregistrée

Cette erreur se produit lorsque vous essayez d'utiliser une fonction qu'Athena ne prend pas en charge. Pour connaître la liste des fonctions prises en charge par Athena, consultezFonctions dans Amazon Athena ou exécutez l'instruction `SHOW FUNCTIONS` dans l'éditeur de requête. Vous pouvez également écrire votre propre fonction définie par l'utilisateur (UDF). Pour plus d'informations, consultez la rubrique Comment résoudre l'erreur de syntaxe « fonction non enregistrée » dans Athena ? dans le Centre de connaissances AWS .

Exceptions GENERIC_INTERNAL_ERROR

Les exceptions `GENERIC_INTERNAL_ERROR` peuvent avoir plusieurs causes, notamment les suivantes :

- GENERIC_INTERNAL_ERROR : NULL – Vous pouvez voir cette exception dans le cadre de l'une des conditions suivantes :
  - Vous avez une inadéquation de schéma entre le type de données d'une colonne dans la définition de table et le type de données réel du jeu de données.
  - Vous exécutez une requête `CREATE TABLE AS SELECT` (CTAS) dont la syntaxe est imprécise.
- GENERIC_INTERNAL_ERROR : le générateur parent est nul. Cette exception peut se produire lorsque vous interrogez une table contenant des colonnes de type de données et que vous utilisez la `array` bibliothèque OpenCSV. SerDe Le format OpenCSVSerDe ne prend pas en charge le type de données `array`.
- GENERIC_INTERNAL_ERROR: Value exceeds MAX_INT (GENERIC_INTERNAL_ERROR : la valeur dépasse MAX_INT) : cette exception peut s'afficher lorsque la colonne de données source est définie avec le type de données `INT` et a une valeur numérique supérieure à 2 147 483 647.
- GENERIC_INTERNAL_ERROR : Value exceeds MAX_BYTE (GENERIC_INTERNAL_ERROR : la valeur dépasse MAX_BYTE) : cette exception peut s'afficher lorsque la valeur numérique de la colonne de données source dépasse la taille autorisée pour le type de données `BYTE`. Le type de données `BYTE` est équivalent à `TINYINT`. `TINYINT` est un entier signé de 8 bits au format de complément à deux avec une valeur minimum de -128 et une valeur maximum de 127.
- GENERIC_INTERNAL_ERROR: Number of partition values does not match number of filters (GENERIC_INTERNAL_ERROR : le nombre de valeurs de partition ne correspond pas au nombre de filtres) : cette exception peut s'afficher si vous avez des partitions incohérentes sur les données Amazon Simple Storage Service (Amazon S3). Les partitions peuvent être incohérentes dans le cadre de l'une des conditions suivantes :
  - Les partitions sur Simple Storage Service (Amazon S3) ont changé (exemple : de nouvelles partitions ont été ajoutées).
  - Le nombre de colonnes de partition de la table ne correspond pas à celui des métadonnées de partition.

Pour des informations plus détaillées sur chacune de ces erreurs, consultez Comment résoudre l'erreur « GENERIC_INTERNAL_ERROR » lorsque j'interroge une table dans Amazon Athena ? dans le AWS Knowledge Center.

Le nombre de groupes correspondants ne correspond pas au nombre de colonnes

Cette erreur se produit lorsque vous utilisez le SerDe Régex SerDe dans une instruction CREATE TABLE et que le nombre de groupes de correspondance regex ne correspond pas au nombre de colonnes que vous avez spécifié pour la table. Pour plus d'informations, consultez Comment résoudre l' RegexSerDe erreur « le nombre de groupes correspondants ne correspond pas au nombre de colonnes » dans Amazon Athena ? dans le AWS Knowledge Center.

queryString ne satisfait pas à la contrainte : le membre doit avoir une longueur inférieure ou égale à 262144

La longueur maximale de la chaîne de requête dans Athena (262 144 octets) n'est pas un quota ajustable. AWS Support vous ne pouvez pas augmenter le quota pour vous, mais vous pouvez contourner le problème en divisant les longues requêtes en requêtes plus petites. Pour plus d'informations, consultez la rubrique Comment augmenter la longueur maximale de la chaîne de requête dans Athena ? dans le Centre de connaissances AWS .

SYNTAX_ERROR : impossible de résoudre la colonne

Cette erreur peut se produire lorsque vous interrogez une table créée par un AWS Glue robot d'exploration à partir d'un fichier CSV codé en UTF-8 comportant une marque d'ordre des octets (BOM). AWS Glue ne reconnaît pas les nomenclatures et les transforme en points d'interrogation, ce qu'Amazon Athena ne reconnaît pas. La solution consiste à supprimer le point d'interrogation dans Athena ou dans AWS Glue.

Trop d'arguments pour un appel de fonction

Dans la version 3 du moteur Athena, les fonctions ne peuvent pas comporter plus de 127 arguments. Cette limitation est intentionnelle. Si vous utilisez une fonction comportant plus de 127 paramètres, un message d'erreur semblable au suivant s'affiche :

TOO_MANY_ARGUMENTS : ligne *nnn*:*nn* : trop d'arguments pour l'appel de fonction *function_name*().

Pour résoudre ce problème, utilisez moins de paramètres par appel de fonction.

Problèmes de délai d'expiration des requêtes

Si vous rencontrez des erreurs de temporisation avec vos requêtes Athena, consultez CloudTrail vos journaux. Les requêtes peuvent expirer en raison de la limitation des API Lake Formation AWS

Glue ou de celles de Lake Formation. Lorsque ces erreurs se produisent, les messages d'erreur correspondants peuvent indiquer un problème de délai d'expiration des requêtes plutôt qu'un problème de limitation. Pour résoudre le problème, vous pouvez consulter vos CloudTrail journaux avant de nous contacter AWS Support. Pour plus d'informations, consultez Journaux d'interrogation AWS CloudTrail et Journalisation des appels d'API Amazon Athena avec AWS CloudTrail.

Pour plus d'informations sur les problèmes de délai d'expiration des requêtes liées aux requêtes fédérées lorsque vous appelez l'API `ListTableMetadata`, veuillez consulter Délai d'attente pendant l'appel ListTableMetadata.

Problèmes de limitation

Si vos requêtes dépassent les limites des services dépendants tels qu'Amazon S3,, ou AWS KMS, AWS Glue,, AWS Lambda,, vous pouvez vous attendre aux messages suivants. Pour résoudre ces problèmes, réduisez le nombre d'appels simultanés provenant du même compte.

Service	Message d'erreur
AWS Glue	AWSGlueException: Taux dépassé.
AWS KMS	Vous avez dépassé la vitesse de transfert à laquelle vous pouvez appeler KMS. Réduisez la fréquence de vos appels.
AWS Lambda	Vitesse de transfert dépassée TooManyRequestsException
Simple Storage Service (Amazon S3)	Amazons3Exception : veuillez réduire la vitesse de transfert de votre demande.

Pour plus d'informations sur les moyens d'empêcher la limitation d'Amazon S3 lorsque vous utilisez Athena, consultez Prévention de la limitation Amazon S3.

Vues

Les vues créées dans le shell Apache Hive ne fonctionnent pas dans Athena

En raison de leurs implémentations fondamentalement différentes, les vues créées dans le shell Apache Hive ne sont pas compatibles avec Athena. Pour résoudre ce problème, recréez les vues dans Athena.

La vue est périmée ; elle doit être recréée

Il est possible de recevoir cette erreur si la table qui sous-tend une vue a été modifiée ou supprimée. La résolution consiste à recréer la vue. Pour plus d'informations, voir Comment puis-je résoudre l'erreur « la vue est périmée ; elle doit être recréée » dans Athéna ? dans le AWS Knowledge Center.

Groupes de travail

Pour plus d'informations sur la résolution des problèmes liés aux groupes de travail, consultez Dépannage des groupes de travail.

Ressources supplémentaires

Les pages suivantes fournissent des informations supplémentaires pour la résolution des problèmes liés à Amazon Athena.

- Catalogue d'erreurs Athena
- Service Quotas
- Considérations et limitations relatives aux requêtes SQL dans Amazon Athena
- DDL non prise en charge
- Noms des tables, des bases de données et des colonnes
- Types de données dans Amazon Athena
- SerDe et formats de données pris en charge
- Prise en charge de la compression Athena
- Mots-clés réservés
- Dépannage des groupes de travail

Les AWS ressources suivantes peuvent également vous être utiles :

- Sujets d'Athena dans le centre de connaissances AWS
- Amazon Athena pose des questions sur Re:Post AWS
- Articles relatifs à Athena sur le blog AWS Big Data

La résolution des problèmes nécessite souvent une requête et une recherche itératives par un expert ou par une communauté d'assistants. Si vous continuez à rencontrer des problèmes après avoir essayé les suggestions de cette page, contactez AWS Support (dans le AWS Management Console, cliquez sur Support, Centre de support) ou posez une question sur AWS Re:post en utilisant le tag Amazon Athena.

Catalogue d'erreurs Athena

Athena fournit des informations d'erreur standardisées pour vous aider à comprendre les requêtes ayant échoué et à prendre des mesures après l'échec d'une requête. La fonction `AthenaError` inclut un champ `ErrorCategory` et un champ `ErrorType`. `ErrorCategory` spécifie si la cause de l'échec de la requête est due à une erreur système, à une erreur utilisateur ou à une autre erreur. `ErrorType` fournit des informations plus détaillées concernant la source de l'échec. En combinant les deux champs, vous pouvez mieux comprendre les circonstances et les causes de l'erreur spécifique qui s'est produite.

Catégorie d'erreurs

Le tableau suivant répertorie les valeurs des catégories d'erreur Athena et leur signification.

Catégorie d'erreurs	Source
1	SYSTEM
2	USER
3	OTHER

Référence du type d'erreur

Le tableau suivant répertorie les valeurs des types d'erreur Athena et leur signification.

Error type (Type d'erreur)	Description
0	La requête a épuisé les ressources à ce facteur d'échelle
1	La requête a épuisé les ressources à ce facteur d'échelle
2	La requête a épuisé les ressources à ce facteur d'échelle
3	La requête a épuisé les ressources à ce facteur d'échelle
4	La requête a épuisé les ressources à ce facteur d'échelle
5	La requête a épuisé les ressources à ce facteur d'échelle
6	La requête a épuisé les ressources à ce facteur d'échelle
7	La requête a épuisé les ressources à ce facteur d'échelle
8	La requête a épuisé les ressources à ce facteur d'échelle
100	Erreur interne du service
200	Le moteur de requête a rencontré une erreur interne
201	Le moteur de requête a rencontré une erreur interne
202	Le moteur de requête a rencontré une erreur interne
203	Erreur de pilote
204	Le métastore a rencontré une erreur
205	Le moteur de requête a rencontré une erreur interne
206	La requête a expiré
207	Le moteur de requête a rencontré une erreur interne

Error type (Type d'erreur)	Description
208	Le moteur de requête a rencontré une erreur interne
209	Échec de l'annulation de la requête
210	La requête a expiré
211	Le moteur de requête a rencontré une erreur interne
212	Le moteur de requête a rencontré une erreur interne
213	Le moteur de requête a rencontré une erreur interne
214	Le moteur de requête a rencontré une erreur interne
215	Le moteur de requête a rencontré une erreur interne
216	Le moteur de requête a rencontré une erreur interne
217	Le moteur de requête a rencontré une erreur interne
218	Le moteur de requête a rencontré une erreur interne
219	Le moteur de requête a rencontré une erreur interne
220	Le moteur de requête a rencontré une erreur interne
221	Le moteur de requête a rencontré une erreur interne
222	Le moteur de requête a rencontré une erreur interne
223	Le moteur de requête a rencontré une erreur interne
224	Le moteur de requête a rencontré une erreur interne
225	Le moteur de requête a rencontré une erreur interne
226	Le moteur de requête a rencontré une erreur interne

Error type (Type d'erreur)	Description
227	Le moteur de requête a rencontré une erreur interne
228	Le moteur de requête a rencontré une erreur interne
229	Le moteur de requête a rencontré une erreur interne
230	Le moteur de requête a rencontré une erreur interne
231	Le moteur de requête a rencontré une erreur interne
232	Le moteur de requête a rencontré une erreur interne
233	Erreur Iceberg
234	Erreur Lake Formation
235	Le moteur de requête a rencontré une erreur interne
236	Le moteur de requête a rencontré une erreur interne
237	Erreur de sérialisation
238	Échec du chargement des métadonnées sur Simple Storage Service (Amazon S3)
239	Erreur de persistance générale
240	Échec de l'envoi de la requête
300	Erreur interne du service
301	Erreur interne du service
302	Erreur interne du service
303	Erreur interne du service
400	Erreur interne du service

Error type (Type d'erreur)	Description
401	Échec de l'écriture des résultats de requête dans Simple Storage Service (Amazon S3)
402	Échec de l'écriture des résultats de requête dans Simple Storage Service (Amazon S3)
1 000	Erreur de l'utilisateur
1001	Erreur de données
1 002	Erreur de données
1003	Échec de la tâche DDL
1004	Erreur de schéma
1005	Erreur de sérialisation
1006	Erreur de syntaxe
1007	Erreur de données
1008	Requête rejetée
1009	Échec de la requête
1010	Erreur interne du service
1011	Requête annulée par l'utilisateur
1012	Le moteur de requête a rencontré une erreur interne
1013	Le moteur de requête a rencontré une erreur interne
1014	Requête annulée par l'utilisateur
1100	Argument fourni non valide
1101	Propriété fournie non valide

Error type (Type d'erreur)	Description
1102	Le moteur de requête a rencontré une erreur interne
1103	Propriété de table fournie non valide
1104	Le moteur de requête a rencontré une erreur interne
1105	Le moteur de requête a rencontré une erreur interne
1106	Argument de fonction fourni non valide
1107	Affichage non valide
1108	Échec d'enregistrement de la fonction
1109	Chemin d'accès Simple Storage Service (Amazon S3) fourni introuvable
1110	La table ou la vue fournie n'existe pas
1200	Requête non prise en charge
1201	Décodeur fourni non pris en charge
1202	Type de requête non prise en charge
1300	Erreur générale introuvable
1301	Entité générale introuvable
1302	Fichier introuvable
1303	Fonction ou implémentation de fonction fournie introuvable
1304	Le moteur de requête a rencontré une erreur interne
1305	Le moteur de requête a rencontré une erreur interne
1306	Compartiment Simple Storage Service (Amazon S3) introuvable

Error type (Type d'erreur)	Description
1307	Moteur sélectionné introuvable
1308	Le moteur de requête a rencontré une erreur interne
1400	Erreur de limitation
1401	Échec de la requête en raison d'un AWS Glue étranglement
1402	La requête a échoué en raison d'un trop grand nombre de versions de table dans AWS Glue
1403	Échec de la requête en raison de la limitation Simple Storage Service (Amazon S3)
1404	Échec de la requête en raison de la limitation Amazon Athena
1405	Échec de la requête en raison de la limitation Amazon Athena
1406	Échec de la requête en raison de la limitation Amazon Athena
1 500	Erreur d'autorisation
1501	Erreur d'autorisation Simple Storage Service (Amazon S3)
1602	Dépassement de la limite de capacité réservée. Capacité insuffisante pour exécuter cette requête.
1700	La requête a échoué en raison d'une exception interne à Lake Formation
1701	Echec de la requête en raison d'une exception AWS Glue interne
9999	Erreur interne du service

Exemples de code

Les exemples de cette rubrique utilisent le kit SDK pour Java 2.x comme point de départ pour écrire des applications Athena.

> **Note**
>
> Pour plus d'informations sur la programmation d'Athena à l'aide d'autres AWS SDK spécifiques à un langage, consultez les ressources suivantes :
>
> - AWS Command Line Interface (`athena`)
> - AWS SDK for .NET (`Amazon.Athena.Model`)
> - AWS SDK for C++ (`Aws::Athena::AthenaClient`)
> - AWS SDK for Go (`athena`)
> - AWS SDK for JavaScript version 3 () `AthenaClient`
> - AWS SDK for PHP 3,x () `Aws\Athena`
> - AWS SDK for Python (Boto3) (`Athena.Client`)
> - AWS SDK for Ruby version 3 () `Aws::Athena::Client`

Pour plus d'informations sur l'exécution des exemples de code Java présentés dans cette section, consultez le fichier readme Java d'Amazon Athena sur le référentiel d'exemples de AWS code sur. GitHub Pour la référence de programmation Java pour Athena, voir AthenaClientdans le. AWS SDK for Java 2.x

- Exemples de code Java
  - Constantes
  - Création d'un client pour accéder à Athena
  - Utilisation d'exécutions de requête
    - Démarrage de l'exécution d'une requête
    - Arrêt de l'exécution d'une requête
    - Listage des exécutions de requête
  - Utilisation de requêtes nommées
    - Création d'une requête nommée

- Suppression d'une requête nommée
- Listage des exécutions de requête

> **Note**
>
> Ces exemples utilisent des constantes (par exemple, `ATHENA_SAMPLE_QUERY`) comme chaînes, qui sont définies dans une déclaration de classe `ExampleConstants.java`. Remplacez ces constantes par vos propres chaînes ou constantes définies.

Constantes

La classe `ExampleConstants.java` montre comment interroger une table créée par le tutoriel de Mise en route dans Athena.

```
package aws.example.athena;

public class ExampleConstants {

    public static final int CLIENT_EXECUTION_TIMEOUT = 100000;
    public static final String ATHENA_OUTPUT_BUCKET = "s3://bucketscott2"; // change
 the Amazon S3 bucket name to match
                                                                          // your
 environment
    // Demonstrates how to query a table with a comma-separated value (CSV) table.
    // For information, see
    // https://docs.aws.amazon.com/athena/latest/ug/work-with-data.html
    public static final String ATHENA_SAMPLE_QUERY = "SELECT * FROM scott2;"; // change
 the Query statement to match
                                                                            // your
 environment
    public static final long SLEEP_AMOUNT_IN_MS = 1000;
    public static final String ATHENA_DEFAULT_DATABASE = "mydatabase"; // change the
 database to match your database

}
```

Création d'un client pour accéder à Athena

La classe `AthenaClientFactory.java` montre comment créer et configurer un client Amazon Athena.

```
package aws.example.athena;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.athena.AthenaClient;
import software.amazon.awssdk.services.athena.AthenaClientBuilder;

public class AthenaClientFactory {
    private final AthenaClientBuilder builder = AthenaClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create());

    public AthenaClient createClient() {
        return builder.build();
    }
}
```

Démarrage de l'exécution d'une requête

`StartQueryExample` montre comment envoyer une requête à Athena, attendre que les résultats soient disponibles, puis traiter les résultats.

```
package aws.example.athena;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.athena.AthenaClient;
import software.amazon.awssdk.services.athena.model.QueryExecutionContext;
import software.amazon.awssdk.services.athena.model.ResultConfiguration;
import software.amazon.awssdk.services.athena.model.StartQueryExecutionRequest;
import software.amazon.awssdk.services.athena.model.StartQueryExecutionResponse;
import software.amazon.awssdk.services.athena.model.AthenaException;
import software.amazon.awssdk.services.athena.model.GetQueryExecutionRequest;
import software.amazon.awssdk.services.athena.model.GetQueryExecutionResponse;
import software.amazon.awssdk.services.athena.model.QueryExecutionState;
import software.amazon.awssdk.services.athena.model.GetQueryResultsRequest;
import software.amazon.awssdk.services.athena.model.GetQueryResultsResponse;
import software.amazon.awssdk.services.athena.model.ColumnInfo;
```

```
import software.amazon.awssdk.services.athena.model.Row;
import software.amazon.awssdk.services.athena.model.Datum;
import software.amazon.awssdk.services.athena.paginators.GetQueryResultsIterable;
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 StartQueryExample {

    public static void main(String[] args) throws InterruptedException {
        AthenaClient athenaClient = AthenaClient.builder()
                .region(Region.US_WEST_2)
                .build();

        String queryExecutionId = submitAthenaQuery(athenaClient);
        waitForQueryToComplete(athenaClient, queryExecutionId);
        processResultRows(athenaClient, queryExecutionId);
        athenaClient.close();
    }

    // Submits a sample query to Amazon Athena and returns the execution ID of the
    // query.
    public static String submitAthenaQuery(AthenaClient athenaClient) {
        try {
            // The QueryExecutionContext allows us to set the database.
            QueryExecutionContext queryExecutionContext =
 QueryExecutionContext.builder()
                    .database(ExampleConstants.ATHENA_DEFAULT_DATABASE)
                    .build();

            // The result configuration specifies where the results of the query should
 go.
            ResultConfiguration resultConfiguration = ResultConfiguration.builder()
                    .outputLocation(ExampleConstants.ATHENA_OUTPUT_BUCKET)
                    .build();

            StartQueryExecutionRequest startQueryExecutionRequest =
 StartQueryExecutionRequest.builder()
```

```
                    .queryString(ExampleConstants.ATHENA_SAMPLE_QUERY)
                    .queryExecutionContext(queryExecutionContext)
                    .resultConfiguration(resultConfiguration)
                    .build();

            StartQueryExecutionResponse startQueryExecutionResponse = athenaClient
                    .startQueryExecution(startQueryExecutionRequest);
            return startQueryExecutionResponse.queryExecutionId();

        } catch (AthenaException e) {
            e.printStackTrace();
            System.exit(1);
        }
        return "";
    }

    // Wait for an Amazon Athena query to complete, fail or to be cancelled.
    public static void waitForQueryToComplete(AthenaClient athenaClient, String
queryExecutionId)
            throws InterruptedException {
        GetQueryExecutionRequest getQueryExecutionRequest =
GetQueryExecutionRequest.builder()
                .queryExecutionId(queryExecutionId)
                .build();

        GetQueryExecutionResponse getQueryExecutionResponse;
        boolean isQueryStillRunning = true;
        while (isQueryStillRunning) {
            getQueryExecutionResponse =
athenaClient.getQueryExecution(getQueryExecutionRequest);
            String queryState =
getQueryExecutionResponse.queryExecution().status().state().toString();
            if (queryState.equals(QueryExecutionState.FAILED.toString())) {
                throw new RuntimeException(
                        "The Amazon Athena query failed to run with error message: " +
getQueryExecutionResponse
                                .queryExecution().status().stateChangeReason());
            } else if (queryState.equals(QueryExecutionState.CANCELLED.toString())) {
                throw new RuntimeException("The Amazon Athena query was cancelled.");
            } else if (queryState.equals(QueryExecutionState.SUCCEEDED.toString())) {
                isQueryStillRunning = false;
            } else {
                // Sleep an amount of time before retrying again.
                Thread.sleep(ExampleConstants.SLEEP_AMOUNT_IN_MS);
```

```
            }
            System.out.println("The current status is: " + queryState);
        }
    }

    // This code retrieves the results of a query
    public static void processResultRows(AthenaClient athenaClient, String
 queryExecutionId) {
        try {
            // Max Results can be set but if its not set,
            // it will choose the maximum page size.
            GetQueryResultsRequest getQueryResultsRequest =
 GetQueryResultsRequest.builder()
                    .queryExecutionId(queryExecutionId)
                    .build();

            GetQueryResultsIterable getQueryResultsResults = athenaClient
                    .getQueryResultsPaginator(getQueryResultsRequest);
            for (GetQueryResultsResponse result : getQueryResultsResults) {
                List<ColumnInfo> columnInfoList =
 result.resultSet().resultSetMetadata().columnInfo();
                List<Row> results = result.resultSet().rows();
                processRow(results, columnInfoList);
            }

        } catch (AthenaException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }

    private static void processRow(List<Row> row, List<ColumnInfo> columnInfoList) {
        for (Row myRow : row) {
            List<Datum> allData = myRow.data();
            for (Datum data : allData) {
                System.out.println("The value of the column is " +
 data.varCharValue());
            }
        }
    }
}
```

Arrêt de l'exécution d'une requête

`StopQueryExecutionExample` exécute un exemple de requête, l'arrête immédiatement, puis vérifie son statut pour s'assurer qu'elle a bien été annulée.

```
package aws.example.athena;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.athena.AthenaClient;
import software.amazon.awssdk.services.athena.model.StopQueryExecutionRequest;
import software.amazon.awssdk.services.athena.model.GetQueryExecutionRequest;
import software.amazon.awssdk.services.athena.model.GetQueryExecutionResponse;
import software.amazon.awssdk.services.athena.model.QueryExecutionState;
import software.amazon.awssdk.services.athena.model.AthenaException;
import software.amazon.awssdk.services.athena.model.QueryExecutionContext;
import software.amazon.awssdk.services.athena.model.ResultConfiguration;
import software.amazon.awssdk.services.athena.model.StartQueryExecutionRequest;
import software.amazon.awssdk.services.athena.model.StartQueryExecutionResponse;

/**
 * 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 StopQueryExecutionExample {
    public static void main(String[] args) {
        AthenaClient athenaClient = AthenaClient.builder()
                .region(Region.US_WEST_2)
                .build();

        String sampleQueryExecutionId = submitAthenaQuery(athenaClient);
        stopAthenaQuery(athenaClient, sampleQueryExecutionId);
        athenaClient.close();
    }

    public static void stopAthenaQuery(AthenaClient athenaClient, String
 sampleQueryExecutionId) {
        try {
            StopQueryExecutionRequest stopQueryExecutionRequest =
 StopQueryExecutionRequest.builder()
```

```
                    .queryExecutionId(sampleQueryExecutionId)
                    .build();

            athenaClient.stopQueryExecution(stopQueryExecutionRequest);
            GetQueryExecutionRequest getQueryExecutionRequest =
GetQueryExecutionRequest.builder()
                    .queryExecutionId(sampleQueryExecutionId)
                    .build();

            GetQueryExecutionResponse getQueryExecutionResponse = athenaClient
                    .getQueryExecution(getQueryExecutionRequest);
            if (getQueryExecutionResponse.queryExecution()
                    .status()
                    .state()
                    .equals(QueryExecutionState.CANCELLED)) {

                System.out.println("The Amazon Athena query has been cancelled!");
            }

        } catch (AthenaException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }

    // Submits an example query and returns a query execution Id value
    public static String submitAthenaQuery(AthenaClient athenaClient) {
        try {
            QueryExecutionContext queryExecutionContext =
QueryExecutionContext.builder()
                    .database(ExampleConstants.ATHENA_DEFAULT_DATABASE)
                    .build();

            ResultConfiguration resultConfiguration = ResultConfiguration.builder()
                    .outputLocation(ExampleConstants.ATHENA_OUTPUT_BUCKET)
                    .build();

            StartQueryExecutionRequest startQueryExecutionRequest =
StartQueryExecutionRequest.builder()
                    .queryExecutionContext(queryExecutionContext)
                    .queryString(ExampleConstants.ATHENA_SAMPLE_QUERY)
                    .resultConfiguration(resultConfiguration).build();

            StartQueryExecutionResponse startQueryExecutionResponse = athenaClient
```

```
                    .startQueryExecution(startQueryExecutionRequest);
            return startQueryExecutionResponse.queryExecutionId();

        } catch (AthenaException e) {
            e.printStackTrace();
            System.exit(1);
        }
        return null;

    }
}
```

Listage des exécutions de requête

`ListQueryExecutionsExample` montre comment obtenir une liste d'ID d'exécution de requête.

```
package aws.example.athena;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.athena.AthenaClient;
import software.amazon.awssdk.services.athena.model.AthenaException;
import software.amazon.awssdk.services.athena.model.ListQueryExecutionsRequest;
import software.amazon.awssdk.services.athena.model.ListQueryExecutionsResponse;
import software.amazon.awssdk.services.athena.paginators.ListQueryExecutionsIterable;
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 ListQueryExecutionsExample {
    public static void main(String[] args) {
        AthenaClient athenaClient = AthenaClient.builder()
                .region(Region.US_WEST_2)
                .build();

        listQueryIds(athenaClient);
        athenaClient.close();
    }
```

```
    public static void listQueryIds(AthenaClient athenaClient) {
        try {
            ListQueryExecutionsRequest listQueryExecutionsRequest =
 ListQueryExecutionsRequest.builder().build();
            ListQueryExecutionsIterable listQueryExecutionResponses = athenaClient
                    .listQueryExecutionsPaginator(listQueryExecutionsRequest);
            for (ListQueryExecutionsResponse listQueryExecutionResponse :
 listQueryExecutionResponses) {
                List<String> queryExecutionIds =
 listQueryExecutionResponse.queryExecutionIds();
                System.out.println("\n" + queryExecutionIds);
            }

        } catch (AthenaException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

Création d'une requête nommée

`CreateNamedQueryExample` montre comment créer une requête nommée.

```
package aws.example.athena;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.athena.AthenaClient;
import software.amazon.awssdk.services.athena.model.AthenaException;
import software.amazon.awssdk.services.athena.model.CreateNamedQueryRequest;

/**
 * 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 CreateNamedQueryExample {
    public static void main(String[] args) {
```

```
        final String USAGE = """

                Usage:
                    <name>

                Where:
                    name - the name of the Amazon Athena query.\s
                """;

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String name = args[0];
        AthenaClient athenaClient = AthenaClient.builder()
                .region(Region.US_WEST_2)
                .build();

        createNamedQuery(athenaClient, name);
        athenaClient.close();
    }

    public static void createNamedQuery(AthenaClient athenaClient, String name) {
        try {
            // Create the named query request.
            CreateNamedQueryRequest createNamedQueryRequest =
 CreateNamedQueryRequest.builder()
                    .database(ExampleConstants.ATHENA_DEFAULT_DATABASE)
                    .queryString(ExampleConstants.ATHENA_SAMPLE_QUERY)
                    .description("Sample Description")
                    .name(name)
                    .build();

            athenaClient.createNamedQuery(createNamedQueryRequest);
            System.out.println("Done");

        } catch (AthenaException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

Suppression d'une requête nommée

`DeleteNamedQueryExample` montre comment supprimer une requête nommée en utilisant l'ID de la requête nommée.

```
package aws.example.athena;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.athena.AthenaClient;
import software.amazon.awssdk.services.athena.model.DeleteNamedQueryRequest;
import software.amazon.awssdk.services.athena.model.AthenaException;
import software.amazon.awssdk.services.athena.model.CreateNamedQueryRequest;
import software.amazon.awssdk.services.athena.model.CreateNamedQueryResponse;

/**
 * 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 DeleteNamedQueryExample {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
                    <name>

                Where:
                    name - the name of the Amazon Athena query.\s
                """;

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String name = args[0];
        AthenaClient athenaClient = AthenaClient.builder()
                .region(Region.US_WEST_2)
                .build();
```

```
        String sampleNamedQueryId = getNamedQueryId(athenaClient, name);
        deleteQueryName(athenaClient, sampleNamedQueryId);
        athenaClient.close();
    }

    public static void deleteQueryName(AthenaClient athenaClient, String
 sampleNamedQueryId) {
        try {
            DeleteNamedQueryRequest deleteNamedQueryRequest =
 DeleteNamedQueryRequest.builder()
                    .namedQueryId(sampleNamedQueryId)
                    .build();

            athenaClient.deleteNamedQuery(deleteNamedQueryRequest);

        } catch (AthenaException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static String getNamedQueryId(AthenaClient athenaClient, String name) {
        try {
            CreateNamedQueryRequest createNamedQueryRequest =
 CreateNamedQueryRequest.builder()
                    .database(ExampleConstants.ATHENA_DEFAULT_DATABASE)
                    .queryString(ExampleConstants.ATHENA_SAMPLE_QUERY)
                    .name(name)
                    .description("Sample description")
                    .build();

            CreateNamedQueryResponse createNamedQueryResponse =
 athenaClient.createNamedQuery(createNamedQueryRequest);
            return createNamedQueryResponse.namedQueryId();

        } catch (AthenaException e) {
            e.printStackTrace();
            System.exit(1);
        }
        return null;
    }
}
```

Listage des requêtes nommées

`ListNamedQueryExample` montre comment obtenir une liste d'ID de requête nommée.

```
package aws.example.athena;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.athena.AthenaClient;
import software.amazon.awssdk.services.athena.model.AthenaException;
import software.amazon.awssdk.services.athena.model.ListNamedQueriesRequest;
import software.amazon.awssdk.services.athena.model.ListNamedQueriesResponse;
import software.amazon.awssdk.services.athena.paginators.ListNamedQueriesIterable;
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 ListNamedQueryExample {
    public static void main(String[] args) {
        AthenaClient athenaClient = AthenaClient.builder()
                .region(Region.US_WEST_2)
                .build();

        listNamedQueries(athenaClient);
        athenaClient.close();
    }

    public static void listNamedQueries(AthenaClient athenaClient) {
        try {
            ListNamedQueriesRequest listNamedQueriesRequest =
 ListNamedQueriesRequest.builder()
                    .build();

            ListNamedQueriesIterable listNamedQueriesResponses = athenaClient
                    .listNamedQueriesPaginator(listNamedQueriesRequest);
            for (ListNamedQueriesResponse listNamedQueriesResponse :
 listNamedQueriesResponses) {
                List<String> namedQueryIds = listNamedQueriesResponse.namedQueryIds();
```

```
                System.out.println(namedQueryIds);
            }

        } catch (AthenaException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

Utilisation d'Apache Spark dans Amazon Athena

Amazon Athena facilite l'exécution interactive de l'analyse et de l'exploration des données à l'aide d'Apache Spark sans qu'il soit nécessaire de planifier, de configurer ou de gérer les ressources. Exécuter des applications Apache Spark sur Athena signifie soumettre du code Spark pour traitement et recevoir directement les résultats sans avoir besoin de configuration supplémentaire. Vous pouvez utiliser l'expérience simplifiée du bloc-notes dans la console Amazon Athena pour développer des applications Apache Spark en utilisant Python ou des API de bloc-notes Athena. Apache Spark fonctionne sur Amazon Athena sans serveur et offre une mise à l'échelle automatique et à la demande qui permet d'obtenir un calcul instantané pour répondre à l'évolution des volumes de données et des exigences de traitement.

Amazon Athena offre les fonctionnalités suivantes :

- Utilisation de la console – Soumettez vos applications Spark à partir de la console Amazon Athena.
- Création de scripts – Créez et déboguez rapidement et de manière interactive des applications Apache Spark en Python.
- Dimensionnement dynamique – Amazon Athena détermine automatiquement les ressources de calcul et de mémoire nécessaires à l'exécution d'une tâche et adapte en permanence ces ressources en conséquence jusqu'aux maximums que vous spécifiez. Ce dimensionnement dynamique réduit le coût sans affecter la vitesse.
- Expérience avec les blocs-notes – Utilisez l'éditeur de bloc-notes Athena pour créer, modifier et exécuter des calculs à l'aide d'une interface familière. Les blocs-notes Athena sont compatibles avec les blocs-notes Jupyter et contiennent une liste de cellules qui sont exécutées dans l'ordre sous forme de calculs. Le contenu des cellules peut inclure du code, du texte, du Markdown, des mathématiques, des diagrammes et des médias enrichis.

Pour plus d'informations, consultez les sections Exécuter Spark SQL sur Amazon Athena Spark et Explorez votre lac de données à l'aide d'Amazon Athena pour Apache Spark sur AWS le blog Big Data.

Considérations et restrictions

- Actuellement, Amazon Athena pour Apache Spark est disponible dans les Régions AWS suivantes :
  - Asie-Pacifique (Mumbai)

  - Asie-Pacifique (Singapour)
  - Asie-Pacifique (Sydney)
  - Asie-Pacifique (Tokyo)
  - Europe (Francfort)
  - Europe (Irlande)
  - USA Est (Virginie du Nord)
  - USA Est (Ohio)
  - USA Ouest (Oregon)
- AWS Lake Formation n'est pas pris en charge.
- Les tables qui utilisent la projection de partitions ne sont pas prises en charge.
- Les groupes de travail compatibles avec Apache Spark peuvent utiliser l'éditeur de bloc-notes Athena, mais pas l'éditeur de requêtes Athena. Seuls les groupes de travail Athena SQL peuvent utiliser l'éditeur de requêtes Athena.
- Les requêtes de vue inter-moteurs ne sont pas prises en charge. Les vues créées par Athena SQL ne sont pas interrogeables par Athena pour Spark. Les vues des deux moteurs étant implémentées différemment, elles ne sont pas compatibles pour une utilisation inter-moteurs.
- MLLib (bibliothèque d'apprentissage automatique Apache Spark) et le `pyspark.ml` package ne sont pas pris en charge. Pour obtenir la liste des bibliothèques Python prises en charge, voir Liste des bibliothèques Python préinstallées.
- Actuellement, n'`pip install`est pas pris en charge dans les sessions Athena pour Spark.
- Une seule session active par bloc-notes est autorisée.
- Lorsque plusieurs utilisateurs utilisent la console pour ouvrir une session existante dans un groupe de travail, ils accèdent au même bloc-notes. Pour éviter toute confusion, n'ouvrez que les sessions que vous créez vous-même.
- Les domaines d'hébergement pour les applications Apache Spark que vous pouvez utiliser avec Amazon Athena (par exemple `analytics-gateway.us-east-1.amazonaws.com`) sont enregistrés dans la liste des suffixes publics (PSL) Internet. Si vous devez définir des cookies sensibles dans vos domaines, nous vous recommandons d'utiliser des cookies avec un préfixe `__Host-` pour protéger votre domaine contre les tentatives CSRF (cross-site request forgery). Pour plus d'informations, veuillez consulter la page Set-Cookie de la documentation pour les développeurs de Mozilla.org (langue française non garantie).
- Pour plus d'informations sur la résolution des problèmes liés aux blocs-notes, sessions et groupes de travail Spark dans Athena, voir Résolution des problèmes liés à Athena pour Spark.

Démarrage avec Apache Spark sur Amazon Athena

Pour commencer à utiliser Apache Spark sur Amazon Athena, vous devez d'abord créer un groupe de travail compatible avec Spark. Après être passé dans le groupe de travail, vous pouvez créer un bloc-notes ou ouvrir un bloc-notes existant. Lorsque vous ouvrez un bloc-notes dans Athena, une nouvelle session est automatiquement lancée pour celui-ci et vous pouvez travailler avec lui directement dans l'éditeur de blocs-notes d'Athena.

> **Note**
>
> Assurez-vous de créer un groupe de travail compatible avec Spark avant de tenter de créer un bloc-notes.

Création d'un groupe de travail compatible avec Spark dans Athena

Vous pouvez utiliser des groupes de travail dans Athena pour regrouper des utilisateurs, des équipes, des applications ou des charges de travail, et pour suivre les coûts. Pour utiliser Apache Spark dans Amazon Athena, vous devez créer un groupe de travail Amazon Athena qui utilise un moteur Spark.

> **Note**
>
> Les groupes de travail compatibles avec Apache Spark peuvent utiliser l'éditeur de bloc-notes Athena, mais pas l'éditeur de requêtes Athena. Seuls les groupes de travail Athena SQL peuvent utiliser l'éditeur de requêtes Athena.

Créer un groupe de travail compatible avec Spark dans Athena

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le panneau de navigation, choisissez Workgroups (Groupes de travail).
4. Sur la page Workgroups (Groupes de travail), choisissez Create workgroup (Créer un groupe de travail).
5. Dans le champ Workgroup name (Nom du groupe de travail), saisissez un nom pour votre groupe de travail Apache Spark.
6. (Facultatif) Dans le champ Description, saisissez une description de votre groupe de travail.
7. Dans le champ Analytics engine (Moteur d'analyse), choisissez Apache Spark.

> **Note**
>
> Une fois que vous avez créé un groupe de travail, le type de moteur d'analyse du groupe de travail ne peut pas être modifié. Par exemple, un groupe de travail du moteur Athena version 3 ne peut pas être remplacé par un groupe de travail PySpark du moteur version 3.

8. Pour les besoins de ce tutoriel, sélectionnez Turn on example notebook (Activer l'exemple de bloc-notes). Cette fonctionnalité facultative ajoute un exemple de bloc-notes portant le nom `example-notebook-`*`random_string`* de votre groupe de travail et ajoute les autorisations AWS Glue associées que le bloc-notes utilise pour créer, afficher et supprimer des bases de données et des tables spécifiques de votre compte, ainsi que des autorisations de lecture dans Amazon S3 pour l'exemple de jeu de données. Pour voir les autorisations ajoutées, choisissez View additional permissions details (Afficher les détails des autorisations supplémentaires).

> **Note**
>
> L'exécution de l'exemple de bloc-notes peut entraîner des coûts supplémentaires.

9. Pour Additional configurations (Configurations supplémentaires), effectuez l'une des opérations suivantes :

   - Utilisez le paramètre Use defaults (Utiliser les valeurs par défaut). Cette option est l'option par défaut et vous aide à démarrer avec votre groupe de travail compatible Spark. Avec cette option, Athena crée pour vous un rôle IAM et un emplacement des résultats de calcul dans Amazon S3. Le nom du rôle IAM et l'emplacement du compartiment S3 à créer s'affichent dans la case située sous le titre Additional configurations (Configurations supplémentaires).
   - Désactivez le paramètre Use defaults (Utiliser les valeurs par défaut), puis suivez les étapes de la rubrique Spécification de vos propres configurations de groupe de travail pour configurer manuellement votre groupe de travail.

10. (Facultatif) Étiquettes – Utilisez cette option pour ajouter des étiquettes à votre groupe de travail. Pour plus d'informations, consultez Étiquetage des ressources Athena.
11. Choisissez Créer un groupe de travail. Un message vous informe que le groupe de travail a été créé avec succès et qu'il apparaît dans la liste des groupes de travail.

Spécification de vos propres configurations de groupe de travail

Si vous souhaitez spécifier votre propre rôle IAM et l'emplacement des résultats de calcul pour votre bloc-notes, suivez les étapes de cette rubrique. Si vous avez sélectionné Use defaults (Utiliser les valeurs par défaut) pour l'option Additional configurations (Configurations supplémentaires), ignorez cette rubrique et passez directement à Ouverture de l'explorateur de blocs-notes et changement de groupe de travail .

La procédure suivante suppose que vous avez terminé les étapes 1 à 9 de la procédure Créer un groupe de travail compatible avec Spark dans Athena décrite à la rubrique précédente.

Spécifier vos propres configurations de groupe de travail

1. Si vous souhaitez créer ou utiliser votre propre rôle IAM ou configurer le chiffrement des blocs-notes, développez la configuration des rôles IAM.

   - Pour Service Role (Fonction du service) sélectionnez l'une des options suivantes :

- Create a service role (Créer une fonction du service) – Choisissez cette option pour qu'Athena crée une fonction du service pour vous. Pour voir les autorisations accordées par le rôle, choisissez View permission details (Afficher les détails des autorisations).
- Choose an existing service role (Choisir une fonction du service existante) – Dans le menu déroulant, choisissez un rôle existant. Le rôle que vous choisissez doit inclure les autorisations de la première option. Pour plus d'informations sur les autorisations pour les groupes de travail avec bloc-notes, voir Résolution des problèmes liés aux groupes de travail compatibles avec Spark.

- Pour Notebook and calculation code encryption key management (Gestion des clés de chiffrement du bloc-notes et du code de calcul), choisissez l'une des options suivantes :
  - Détenue par Amazon Athena — La AWS KMS clé est détenue et gérée par Amazon Athena. Aucun frais supplémentaire ne vous est facturé pour l'utilisation de cette clé.
  - A symmetric key stored in your account, owned and managed by you (Une clé symétrique stockée dans votre compte, détenue et gérée par vous) – Pour cette option, effectuez l'une des opérations suivantes :
    - Pour utiliser une clé existante, utilisez le champ de recherche pour choisir AWS KMS ou saisir un ARN de clé.
    - Pour créer une clé dans la AWS KMS console, choisissez Créer une AWS KMS clé. Votre rôle d'exécution doit avoir l'autorisation d'utiliser la clé que vous créez.

> **Important**
>
> Lorsque vous modifiez la AWS KMS key pour un groupe de travail, les blocs-notes gérés avant la mise à jour font toujours référence à l'ancienne clé KMS. Les blocs-notes gérés après la mise à jour utilisent la nouvelle clé KMS. Pour mettre à jour les anciens blocs-notes afin de faire référence à la nouvelle clé KMS, exportez puis importez chacun des anciens blocs-notes. Si vous supprimez l'ancienne clé KMS avant de mettre à jour les références des anciens blocs-notes avec la nouvelle clé KMS, les anciens blocs-notes ne sont plus déchiffrables et ne peuvent pas être récupérés.
>
> Ce comportement s'applique également aux mises à jour des alias, qui sont des noms conviviaux pour les clés KMS. Lorsque vous mettez à jour un alias de clé KMS pour pointer vers une nouvelle clé KMS, les blocs-notes gérés avant la mise à jour de l'alias font toujours référence à l'ancienne clé KMS, et les blocs-notes gérés après la mise à

jour de l'alias utilisent la nouvelle clé KMS. Tenez compte des points suivants avant de mettre à jour vos clés ou alias KMS.

2. Si vous voulez spécifier vos propres paramètres de résultat de calcul, développez Calculation result settings (Paramètres de résultat de calcul), puis choisissez parmi les options suivantes.

   - Create a new S3 bucket (Créer un nouveau compartiment S3) – Cette option crée un compartiment Amazon S3 dans votre compte pour les résultats de vos calculs. Le nom du compartiment a le format *`account_id`*`-`*`region`*`-athena-results-bucket-`*`alphanumeric_id`* et utilise les paramètres ACL désactivés, l'accès public bloqué, la gestion des versions désactivée et le propriétaire du compartiment imposé.
   - Choose an existing S3 location (Choisir un emplacement S3 existant) – Pour cette option, procédez comme suit :
     - Saisissez le chemin d'un emplacement S3 existant dans la zone de recherche, ou choisissez Browse S3 (Parcourir S3) pour choisir un compartiment dans une liste.

       > **Note**
       >
       > Lorsque vous sélectionnez un emplacement existant dans Amazon S3, n'ajoutez pas de barre oblique (`/`) à cet emplacement. Ainsi, le lien vers l'emplacement des résultats du calcul sur la page des détails du calcul pointe vers un répertoire incorrect. Si cela se produit, modifiez l'emplacement des résultats du groupe de travail pour supprimer la barre oblique de fin de ligne.

     - (Facultatif) Choisissez View (Afficher) pour ouvrir la page Buckets (Compartiments) de la console Amazon S3 où vous pouvez voir plus d'informations sur le compartiment existant que vous avez choisi.
     - (Facultatif) Dans le champ Propriétaire attendu du bucket, entrez l'ID de AWS compte que vous pensez être le propriétaire du bucket d'emplacement de sortie des résultats de votre requête. Nous vous recommandons de choisir cette option comme mesure de sécurité supplémentaire lorsque cela est possible. Si l'ID du compte du propriétaire du compartiment ne correspond pas à l'ID que vous avez spécifié, les tentatives de sortie vers le compartiment échoueront. Pour obtenir des informations détaillées, consultez Vérification de la propriété du compartiment avec la condition de propriétaire du compartiment dans le Guide de l'utilisateur Simple Storage Service (Amazon S3).
     - (Facultatif) Sélectionnez Assign bucket owner full control over query results (Attribuer au propriétaire du compartiment le contrôle total des résultats de la requête) si l'emplacement

des résultats de votre calcul est détenu par un autre compte et si vous voulez accorder à cet autre compte le contrôle total des résultats de votre requête.

3. (Facultatif) Sélectionnez Encrypt calculation results (Chiffrer les résultats du calcul), puis choisissez l'une des options suivantes :

   - SSE_S3 – Il s'agit d'une clé de chiffrement côté serveur gérée par S3.
   - SSE_KMS – Une clé que vous fournissez. Pour Choisir une AWS KMS clé, vous pouvez choisir l'une des options suivantes :
     - Utiliser une clé AWS détenue — Utilisez une clé qui AWS possède et gère pour vous.
     - Choisir une autre AWS KMS touche (avancée) : choisissez ou créez une clé.
       - Pour utiliser une clé existante, utilisez le champ de recherche pour choisir AWS KMS ou saisir un ARN de clé.
       - Pour créer une clé dans la console KMS, choisissez Create an AWS KMS key. Après avoir créé la clé dans la console KMS, retournez à la page Créer un groupe de travail dans la console Athena, puis utilisez le champ de recherche Choisissez AWS KMS une clé ou entrez un ARN pour choisir la clé que vous venez de créer.

4. (Facultatif) Autres paramètres : développez cette option pour activer ou désactiver l'option Publier CloudWatch les métriques pour le groupe de travail. Ce champ est sélectionné par défaut. Pour plus d'informations, consultez Surveillance des calculs Apache Spark à l'aide des métriques CloudWatch.

5. (Facultatif) Étiquettes – Utilisez cette option pour ajouter des étiquettes à votre groupe de travail. Pour plus d'informations, consultez Étiquetage des ressources Athena.

6. Choisissez Créer un groupe de travail. Un message vous informe que le groupe de travail a été créé avec succès et qu'il apparaît dans la liste des groupes de travail.

Ouverture de l'explorateur de blocs-notes et changement de groupe de travail

Pour pouvoir utiliser le groupe de travail compatible avec Spark que vous venez de créer, vous devez passer au groupe de travail. Pour changer de groupe de travail compatible avec Spark, vous pouvez utiliser l'option Workgroup (Groupe de travail) dans l'explorateur ou l'éditeur de blocs-notes.

> **Note**
>
> Avant de commencer, vérifiez que votre navigateur ne bloque pas les cookies tiers. Tout navigateur qui bloque les cookies tiers par défaut ou en tant que paramètre activé par l'utilisateur empêchera le lancement des bloc-notes. Pour en savoir plus sur la gestion des cookies, voir :
>
> - Chrome
> - Firefox
> - Safari

Ouvrir l'explorateur de blocs-notes et changer de groupe de travail

1. Dans le volet de navigation, choisissez Notebook explorer (Explorateur de bloc-notes).
2. Utilisez l'option Workgroup (Groupe de travail) en haut à droite de la console pour choisir le groupe de travail compatible avec Spark que vous avez créé. L'exemple de bloc-notes est affiché dans la liste des blocs-notes.

   Vous pouvez utiliser l'explorateur de bloc-notes de la manière suivante :

   - Choisissez le nom associé d'un bloc-notes pour ouvrir le bloc-notes dans une nouvelle session.
   - Pour renommer, supprimer ou exporter votre bloc-notes, utilisez le menu Actions.
   - Pour importer un fichier de bloc-notes, sélectionnez Import file (Importer un fichier).
   - Pour créer un bloc-notes, sélectionnez Create notebook (Créer un bloc-notes).

Exécution de l'exemple de bloc-notes

L'exemple de bloc-notes interroge les données d'un jeu de données disponible publiquement sur les trajets en taxi dans la ville de New York. Le bloc-notes contient des exemples qui montrent comment utiliser Spark DataFrames, Spark SQL et le AWS Glue Data Catalog.

Exécuter l'exemple de bloc-notes

1. Dans l'explorateur de blocs-notes, choisissez le nom associé à l'exemple de bloc-notes.

Cela démarre une session de bloc-notes avec les paramètres par défaut et ouvre le bloc-notes dans l'éditeur de blocs-notes. Un message vous informe qu'une nouvelle session Apache Spark a été lancée en utilisant les paramètres par défaut (20 DPU maximum).

2. Pour exécuter les cellules dans l'ordre et observer les résultats, cliquez une fois sur le bouton Run (Exécuter) pour chaque cellule du bloc-notes.

   - Faites défiler vers le bas pour voir les résultats et faire apparaître de nouvelles cellules.
   - Pour les cellules qui comportent un calcul, une barre de progression indique le pourcentage achevé, le temps écoulé et le temps restant.
   - L'exemple de bloc-notes crée une base de données et une table dans votre compte. La dernière cellule les supprime lors d'une étape de nettoyage.

> **Note**
>
> Si vous modifiez les noms de dossier, de table ou de base de données dans l'exemple de bloc-notes, assurez-vous que ces modifications sont reflétées dans les rôles IAM que vous utilisez. Sinon, le bloc-notes risque de ne pas fonctionner en raison d'autorisations insuffisantes.

Modification des détails de la session

Après avoir démarré une session de bloc-notes, vous pouvez modifier les détails de la session, comme le format de table, le chiffrement, le délai d'inactivité de la session et le nombre maximum d'unités de traitement de données (DPU) que vous souhaitez utiliser simultanément. Une DPU est une mesure relative de la puissance de traitement consistant en 4 vCPU de capacité de calcul et 16 Go de mémoire.

Modifier les détails de la session

1. Dans l'éditeur de bloc-notes, dans le menu Session en haut à droite, choisissez Edit session (Modifier la session).
2. Dans la boîte de dialogue Modifier les détails de la session, dans la section Propriétés Spark, choisissez ou saisissez des valeurs pour les options suivantes :

- Format de table supplémentaire : choisissez Linux Foundation Delta Lake, Apache Hudi, Apache Iceberg ou Personnalisé.
  - Pour les options de table Delta, Hudi ou Iceberg, les propriétés de table requises pour le format de table correspondant vous sont automatiquement fournies dans les options Modifier dans la table et Modifier dans JSON. Pour plus d'informations sur l'utilisation de ces formats de table, consultez Utilisation de formats de table autres que Hive dans Amazon Athena pour Apache Spark.
  - Pour ajouter ou supprimer des propriétés de table pour le type de table personnalisé ou pour d'autres types de table, utilisez les options Modifier dans la table et Modifier dans JSON.
  - Pour l'option Modifier dans la table, choisissez Ajouter une propriété pour ajouter une propriété, ou choisissez Supprimer pour supprimer une propriété. Utilisez les champs Clé et Valeur pour saisir les noms des propriétés et leurs valeurs.
  - Pour l'option Modifier dans JSON, utilisez l'éditeur de texte JSON pour modifier directement la configuration.
    - Choisissez Copier pour copier le texte JSON dans le presse-papier.
    - Choisissez Effacer pour supprimer tout le texte de l'éditeur JSON.
    - Choisissez l'icône des paramètres (engrenage) pour configurer l'encapsulage des lignes ou choisissez un thème de couleur pour l'éditeur JSON.
- Activer le chiffrement Spark : sélectionnez cette option pour chiffrer les données écrites sur le disque et envoyées via les nœuds du réseau Spark. Pour plus d'informations, consultez Activation du chiffrement Apache Spark.

3. Dans la section Paramètres de la session, choisissez ou saisissez des valeurs pour les options suivantes :

- Session idle timeout (Délai d'inactivité de la session) – Choisissez ou saisissez une valeur comprise entre 1 et 480 minutes. La valeur par défaut est de 20.
- Coordinator size (Taille du coordinateur) – Le coordinateur est un exécuteur spécial qui orchestre le travail de traitement et gère les autres exécuteurs d'une session de bloc-notes. Actuellement, 1 DPU est la valeur par défaut et la seule valeur possible.
- Executor size (Taille de l'exécuteur) – L'exécuteur est la plus petite unité de calcul qu'une session de bloc-notes peut demander à Athena. Actuellement, 1 DPU est la valeur par défaut et la seule valeur possible.
- Max concurrent value (Valeur maximale de traitement simultané) – Le nombre maximal de DPU pouvant être exécutés simultanément. La valeur par défaut est 20, le minimum est 3 et le

maximum est 60. L'augmentation de cette valeur n'entraîne pas automatiquement l'allocation de ressources supplémentaires, mais Athena tentera d'allouer jusqu'au maximum spécifié lorsque la charge de calcul le nécessite et que des ressources sont disponibles.

4. Choisissez Enregistrer.
5. À l'invite Confirm edit (Confirmer la modification), choisissez Confirm (Confirmer).

   Athena enregistre votre bloc-notes et démarre une nouvelle session avec les paramètres que vous avez spécifiés. Une bannière dans l'éditeur de bloc-notes vous informe qu'une nouvelle session a commencé avec les paramètres modifiés.

> **Note**
>
> Athena mémorise les paramètres de votre session pour le bloc-notes. Si vous modifiez les paramètres d'une session et que vous mettez ensuite fin à la session, Athena utilise les paramètres de session que vous avez configurés la prochaine fois que vous démarrez une session pour le bloc-notes.

Affichage des détails de la session et des calculs

Après avoir exécuté le bloc-notes, vous pouvez consulter les détails de votre session et de vos calculs.

Afficher les détails de la session et des calculs

1. Dans le menu Session en haut à droite, choisissez View details (Afficher les détails).
   - L'onglet Current session (Session en cours) affiche des informations sur la session en cours, notamment l'ID de la session, l'heure de création, l'état et le groupe de travail.
   - L'onglet History (Historique) répertorie les ID des sessions précédentes. Pour afficher les détails d'une session précédente, sélectionnez l'onglet History (Historique), puis choisissez un ID de session dans la liste.
   - La section Calculations (Calculs) affiche une liste des calculs effectués au cours de la session.
2. Pour afficher les détails d'un calcul, choisissez l'ID du calcul.
3. Sur la page Calculation details (Détails du calcul), vous pouvez effectuer les opérations suivantes :

- Pour consulter le code du calcul, voir la section Code.
- Pour voir les résultats du calcul, choisissez l'onglet Results (Résultats).
- Pour télécharger les résultats que vous voyez en format texte, sélectionnez Download results (Télécharger les résultats).
- Pour afficher les informations sur les résultats du calcul dans Amazon S3, choisissez View in S3 (Afficher dans S3).

Terminer une session

Mettre fin à une session de bloc-notes

1. Dans l'éditeur de bloc-notes, dans le menu Session en haut à droite, choisissez Terminate (Terminer).
2. À l'invite Confirm session termination (Confirmer la fin de la session), choisissez Confirm (Confirmer). Votre bloc-notes est enregistré et vous êtes renvoyé à l'éditeur de blocs-notes.

> **Note**
>
> La fermeture d'un onglet de bloc-notes dans l'éditeur de bloc-notes ne met pas automatiquement fin à la session d'un bloc-notes actif. Si vous voulez vous assurer que la session est terminée, utilisez l'option Session, Terminate (Terminer).

Création de votre propre bloc-notes

Une fois que vous avez créé un groupe de travail Athena compatible avec Spark, vous pouvez créer votre propre bloc-notes.

Pour créer un bloc-notes

1. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
2. Dans le volet de navigation de la console Athena, choisissez Notebook Explorer (Explorateur de bloc-notes) ou Notebook Editor (Éditeur de bloc-notes).
3. Effectuez l’une des actions suivantes :

- Dans Notebook explorer (Explorateur de blocs-notes), choisissez Create notebook (Créer un bloc-notes).
- Dans Notebook editor (Éditeur de bloc-notes), choisissez Create notebook (Créer un bloc-notes) ou cliquez sur l'icône plus (+) pour ajouter un bloc-notes.

4. Dans la boîte de dialogue Create notebook (Créer un bloc-notes), saisissez un nom dans le champ Notebook name (Nom du bloc-notes).
5. (Facultatif) Développez Propriétés Spark, puis choisissez ou saisissez des valeurs pour les options suivantes :

   - Format de table supplémentaire : choisissez Linux Foundation Delta Lake, Apache Hudi, Apache Iceberg ou Personnalisé.
     - Pour les options de table Delta, Hudi ou Iceberg, les propriétés de table requises pour le format de table correspondant vous sont automatiquement fournies dans les options Modifier dans la table et Modifier dans JSON. Pour plus d'informations sur l'utilisation de ces formats de table, consultez Utilisation de formats de table autres que Hive dans Amazon Athena pour Apache Spark.
     - Pour ajouter ou supprimer des propriétés de table pour le type de table personnalisé ou pour d'autres types de table, utilisez les options Modifier dans la table et Modifier dans JSON.
     - Pour l'option Modifier dans la table, choisissez Ajouter une propriété pour ajouter une propriété, ou choisissez Supprimer pour supprimer une propriété. Utilisez les champs Clé et Valeur pour saisir les noms des propriétés et leurs valeurs.
     - Pour l'option Modifier dans JSON, utilisez l'éditeur de texte JSON pour modifier directement la configuration.
       - Choisissez Copier pour copier le texte JSON dans le presse-papier.
       - Choisissez Effacer pour supprimer tout le texte de l'éditeur JSON.
       - Choisissez l'icône des paramètres (engrenage) pour configurer l'encapsulage des lignes ou choisissez un thème de couleur pour l'éditeur JSON.
   - Activer le chiffrement Spark : sélectionnez cette option pour chiffrer les données écrites sur le disque et envoyées via les nœuds du réseau Spark. Pour plus d'informations, consultez Activation du chiffrement Apache Spark.

6. (Facultatif) Développez Session parameters (Paramètres de session), puis choisissez ou saisissez des valeurs pour les options suivantes :

- Session idle timeout (Délai d'inactivité de la session) – Choisissez ou saisissez une valeur comprise entre 1 et 480 minutes. La valeur par défaut est de 20.
- Coordinator size (Taille du coordinateur) – Le coordinateur est un exécuteur spécial qui orchestre le travail de traitement et gère les autres exécuteurs d'une session de bloc-notes. Actuellement, 1 DPU est la valeur par défaut et la seule valeur possible. La DPU (unité de traitement des données) est une mesure relative de la puissance de traitement qui consiste en une capacité de calcul de 4 vCPU et une mémoire de 16 Go.
- Executor size (Taille de l'exécuteur) – L'exécuteur est la plus petite unité de calcul qu'une session de bloc-notes peut demander à Athena. Actuellement, 1 DPU est la valeur par défaut et la seule valeur possible.
- Max concurrent value (Valeur maximale de traitement simultané) – Le nombre maximal de DPU pouvant être exécutés simultanément. La valeur par défaut est 20 et la valeur maximale est 60. L'augmentation de cette valeur n'entraîne pas automatiquement l'allocation de ressources supplémentaires, mais Athena tentera d'allouer jusqu'au maximum spécifié lorsque la charge de calcul le nécessite et que des ressources sont disponibles.

7. Choisissez Créer. Votre bloc-notes s'ouvre dans une nouvelle session dans l'éditeur de bloc-notes.

Ouverture d'un bloc-notes créé précédemment

Ouvrir un bloc-notes créé précédemment

1. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
2. Dans le volet de navigation de la console Athena, choisissez Notebook editor (Éditeur de bloc-notes) ou Notebook explorer (Explorateur de bloc-notes).
3. Effectuez l’une des actions suivantes :

   - Dans Notebook editor (Éditeur de bloc-notes), choisissez un bloc-notes dans la liste Recent notebooks (Blocs-notes récents) ou Saved notebooks (Blocs-notes enregistrés). Le bloc-notes s'ouvre dans une nouvelle session.
   - Dans Notebook explorer (Explorateur de blocs-notes), choisissez le nom d'un bloc-notes dans la liste. Le bloc-notes s'ouvre dans une nouvelle session.

Pour plus d'informations sur la gestion des fichiers de votre bloc-notes, voir Gestion des fichiers de bloc-notes.

Utilisation des blocs-notes

Vous gérez vos blocs-notes dans l'explorateur de blocs-notes Athena et vous les modifiez et les exécutez dans des sessions à l'aide de l'éditeur de blocs-notes Athena. Vous pouvez configurer l'utilisation des DPU pour vos sessions de bloc-notes en fonction de vos besoins.

Lorsque vous arrêtez un bloc-notes, vous terminez la session associée. Tous les fichiers sont sauvegardés, mais les changements en cours dans les variables, fonctions et classes déclarées sont perdus. Lorsque vous redémarrez le bloc-notes, Athena recharge les fichiers du bloc-notes et vous pouvez réexécuter votre code.

Sessions et calculs

Chaque bloc-notes est associé à un seul noyau Python et exécute du code Python. Un bloc-notes peut contenir une ou plusieurs cellules contenant des commandes. Pour exécuter les cellules d'un bloc-notes, vous devez d'abord créer une session pour le bloc-notes. Les sessions permettent de suivre les variables et l'état des blocs-notes.

L'exécution d'une cellule dans un bloc-notes signifie exécuter un calcul dans la session en cours. Les calculs font progresser l'état du bloc-notes et peuvent effectuer des tâches telles que la lecture depuis Amazon S3 ou l'écriture dans d'autres magasins de données. Tant qu'une session est en cours d'exécution, les calculs utilisent et modifient l'état conservé pour le bloc-notes.

Lorsque vous n'avez plus besoin de l'état, vous pouvez mettre fin à la session. Lorsque vous mettez fin à une session, le bloc-notes est conservé, mais les variables et autres informations d'état sont détruites. Si vous devez travailler sur plusieurs projets en même temps, vous pouvez créer une session pour chaque projet, et les sessions seront indépendantes les unes des autres.

Les sessions disposent d'une capacité de calcul dédiée, mesurée en DPU. Lorsque vous créez une session, vous pouvez lui attribuer un certain nombre de DPU. Les différentes sessions peuvent avoir des capacités différentes en fonction des exigences de la tâche.

Utilisation de l'éditeur de bloc-notes Athena

L'éditeur de bloc-notes Athena est un environnement interactif permettant d'écrire et d'exécuter du code. Les sections suivantes décrivent les caractéristiques de l'environnement.

Mode commande vs mode édition

L'éditeur de bloc-notes possède une interface utilisateur modale : un mode édition pour saisir du texte dans une cellule, et un mode commande pour envoyer des commandes à l'éditeur lui-même, comme copier, coller ou exécuter.

Pour utiliser le mode édition et le mode commande, vous pouvez effectuer les tâches suivantes :

- Pour passer en mode édition, appuyez sur **ENTER** ou choisissez une cellule. Lorsqu'une cellule est en mode édition, elle présente une marge gauche verte.
- Pour passer en mode commande, appuyez sur **ESC** ou cliquez à l'extérieur d'une cellule. Notez que les commandes s'appliquent généralement uniquement à la cellule actuellement sélectionnée, et non à toutes les cellules. Lorsque l'éditeur est en mode commande, la cellule présente une marge gauche bleue.
- En mode commande, vous pouvez utiliser les raccourcis clavier et le menu situé au-dessus de l'éditeur, mais vous ne pouvez pas saisir de texte dans des cellules individuelles.
- Pour sélectionner une cellule, choisissez-la.
- Pour sélectionner toutes les cellules, appuyez sur **Ctrl+A** (Windows) ou sur **Cmd+A** (Mac).

Menu de l'éditeur de bloc-notes

Les icônes du menu situé en haut de l'éditeur de bloc-notes offrent les options suivantes :

- Enregistrer – Permet d'enregistrer l'état actuel du bloc-notes.
- Insérer une cellule en dessous – Permet d'ajouter une nouvelle cellule (vide) en dessous de la cellule actuellement sélectionnée.
- Couper les cellules sélectionnées – Permet de supprimer la cellule sélectionnée de son emplacement actuel et de la copier dans la mémoire.
- Copier les cellules sélectionnées – Permet de copier la cellule sélectionnée dans la mémoire.
- Coller les cellules en dessous – Permet de coller la cellule copiée en dessous de la cellule actuelle.
- Déplacer les cellules sélectionnées vers le haut – Permet de déplacer la cellule actuelle au-dessus de la cellule située au-dessus.
- Déplacer les cellules sélectionnées vers le bas – Permet de déplacer la cellule actuelle sous la cellule située en dessous.
- Exécuter – Permet d'exécuter la cellule actuelle (sélectionnée). La sortie s'affiche immédiatement sous la cellule actuelle.

- Tout exécuter – Permet d'exécuter toutes les cellules du bloc-notes. Le résultat de chaque cellule s'affiche immédiatement sous la cellule.
- Stop (Interrompre le noyau) – Permet d'arrêter le bloc-notes actuel en interrompant le noyau.
- Option de format – Permet de sélectionner le format de cellule, qui peut être l'un des suivants :
  - Code – À utiliser pour le code Python (par défaut).
  - Markdown — À utiliser pour saisir du texte au format Markdown de GitHub style -style. Pour afficher le format Markdown, exécutez la cellule.
  - NBConvert brut – Permet de saisir du texte sous forme non modifiée. Les cellules marquées comme NBConvert brut peuvent être converties dans un format différent, comme le HTML, à l'aide de l'outil de ligne de commande nbconvert de Jupyter.
- Titre – Permet de modifier le niveau de titre de la cellule.
- Palette de commandes – Contient les commandes du bloc-notes Jupyter et leurs raccourcis clavier. Pour plus d'informations sur les raccourcis clavier, voir les rubriques suivantes du présent document.
- Session – Utilisez les options de ce menu pour afficher les détails d'une session, modifier les paramètres de session ou mettre fin à la session.

Raccourcis clavier en mode commande

Voici quelques raccourcis clavier courants du mode commande de l'éditeur de bloc-notes. Ces raccourcis sont disponibles après avoir appuyé sur **ESC** pour passer en mode commande. Pour consulter la liste complète des commandes disponibles dans l'éditeur, appuyez sur **ESC + H**.

Clé	Action
1 - 6	Passer le type de cellule au format Markdown et définir le niveau de titre au numéro saisi
a	Créer une cellule au-dessus de la cellule actuelle
b	Créer une cellule en dessous de la cellule actuelle
c	Copier la cellule actuelle dans la mémoire
d d	Supprimer la cellule actuelle

Clé	Action
h	Afficher l'écran d'aide des raccourcis clavier
j	Descendre d'une cellule
k	Monter d'une cellule
m	Changer le format actuel des cellules en format Markdown
r	Changer le format actuel des cellules en format brut
s	Enregistrer le bloc-notes
v	Coller le contenu de la mémoire sous la cellule actuelle
x	Couper la ou les cellules sélectionnées
y	Changer le format de la cellule en code
z	Annuler
Ctrl+Enter	Exécuter la cellule actuelle et passer en mode commande
Shift+Enter ou **Alt+Enter**	Exécuter la cellule actuelle et créer une nouvelle cellule sous la sortie, puis passer la nouvelle cellule en mode édition
Space	Page avant
Shift+Space	Page arrière
Shift + L	Activer ou désactiver la visibilité des numéros de ligne dans les cellules

Modification des raccourcis clavier du mode commande

L'éditeur de bloc-notes dispose d'une option permettant de personnaliser les raccourcis clavier du mode commande.

Modifier les raccourcis clavier du mode commande

1. Dans le menu de l'éditeur de bloc-notes, choisissez la palette de commandes.
2. Dans la palette de commandes, choisissez la commande Edit command mode keyboard shortcuts (Modifier les raccourcis clavier du mode de commande).
3. Utilisez l'interface Edit command mode shortcuts (Modifier les raccourcis clavier du mode de commande) pour mapper ou remapper les commandes que vous souhaitez sur le clavier.

   Pour voir les instructions de modification des raccourcis clavier du mode commande, faites défiler jusqu'au bas de l'écran Edit command mode shortcuts (Modifier les raccourcis clavier du mode de commande).

Pour plus d'informations sur l'utilisation des commandes magiques dans Athena pour Apache Spark, consultez Utilisation des commandes magiques.

Utilisation des commandes magiques

Les commandes magiques, ou magies, sont des commandes spéciales que vous pouvez exécuter dans une cellule de bloc-notes. Par exemple, `%env` affiche les variables d'environnement dans une session de bloc-notes. Athena prend en charge les fonctions magiques d'IPython 6.0.3.

Cette section présente certaines commandes magiques clés d'Athena pour Apache Spark.

- Pour consulter la liste des commandes magiques d'Athena, exécutez la commande **`%lsmagic`** dans une cellule de bloc-notes.
- Pour plus d'informations sur l'utilisation des magies pour créer des graphiques dans les blocs-notes Athena, consultez Magies pour créer des graphiques de données.
- Pour plus d'informations sur les commandes magiques supplémentaires, consultez Commandes magiques intégrées dans la documentation IPython.

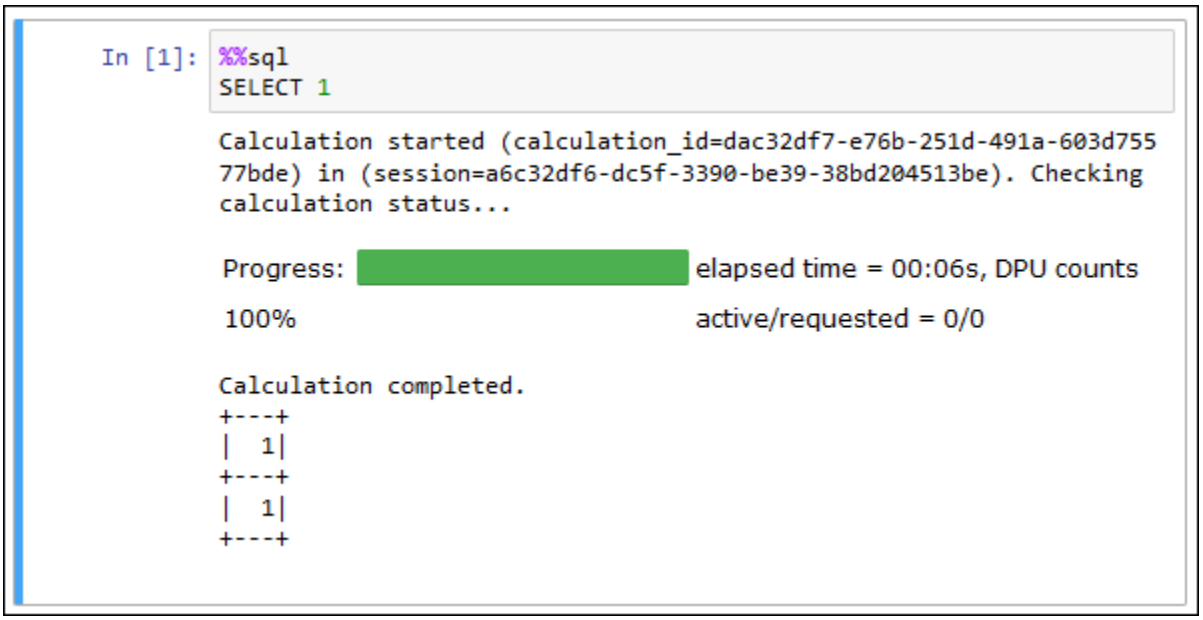
> **Note**
>
> Actuellement, l'exécution de la commande `%pip` échoue. Il s'agit d'un problème connu.

Magies cellulaires

Les magies qui sont écrites sur plusieurs lignes sont précédées d'un double signe de pourcentage (`%%`) et sont appelées fonctions de magie cellulaire ou magies cellulaires.

%%sql

Cette magie cellulaire permet d'exécuter des instructions SQL directement sans avoir à la décorer avec des instructions SQL Spark. La commande affiche également la sortie en appelant implicitement `.show()` sur le cadre de données renvoyé.



La commande `%%sql` tronque automatiquement les sorties de colonne à une largeur de 20 caractères. Actuellement, ce paramètre n'est pas configurable. Pour contourner cette limite, utilisez la syntaxe complète suivante et modifiez les paramètres de la méthode `show` en conséquence.

```
spark.sql("""YOUR_SQL""").show(n=number, truncate=number, vertical=bool)
```

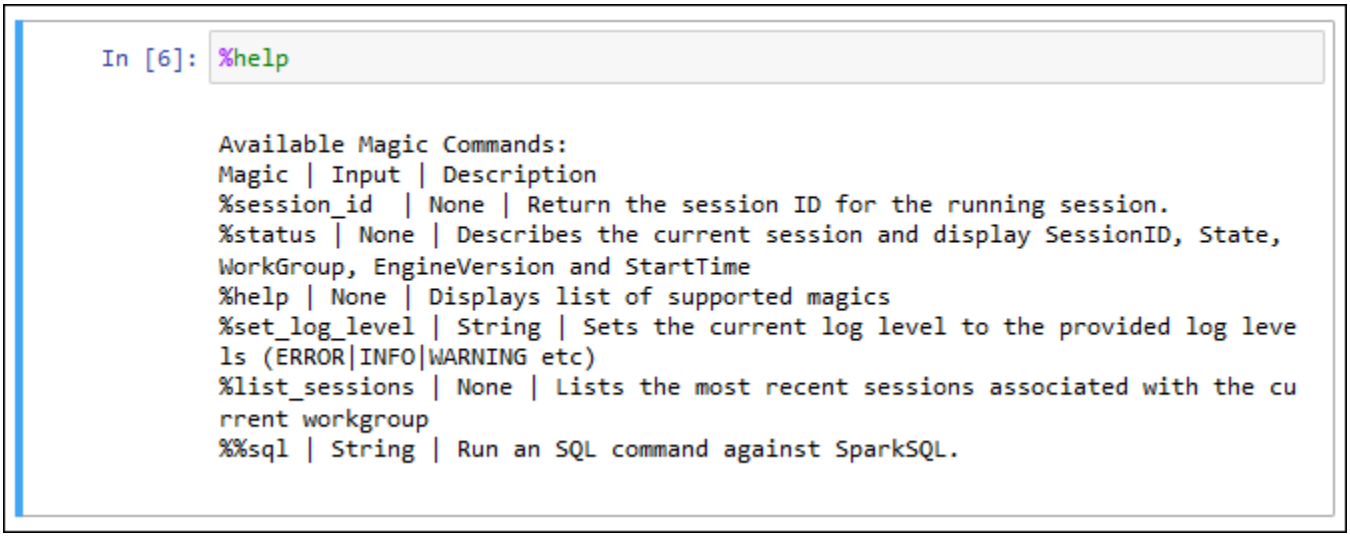
- n `int`, facultatif. Nombre de lignes à afficher.
- tronquer – `bool` ou`int`, facultatif : si `true`, tronque les chaînes de plus de 20 caractères. Lorsqu'il est défini sur un nombre supérieur à 1, tronque les chaînes longues à la longueur spécifiée et aligne les cellules à droite.
- vertical – `bool`, facultatif. Si `true`, imprime les lignes de sortie verticalement (une ligne par valeur de colonne).

Magies linéaires

Les magies qui se trouvent sur une seule ligne sont précédées d'un signe de pourcentage (%) et sont appelées fonctions de magie linéaire ou magies linéaires.

%help

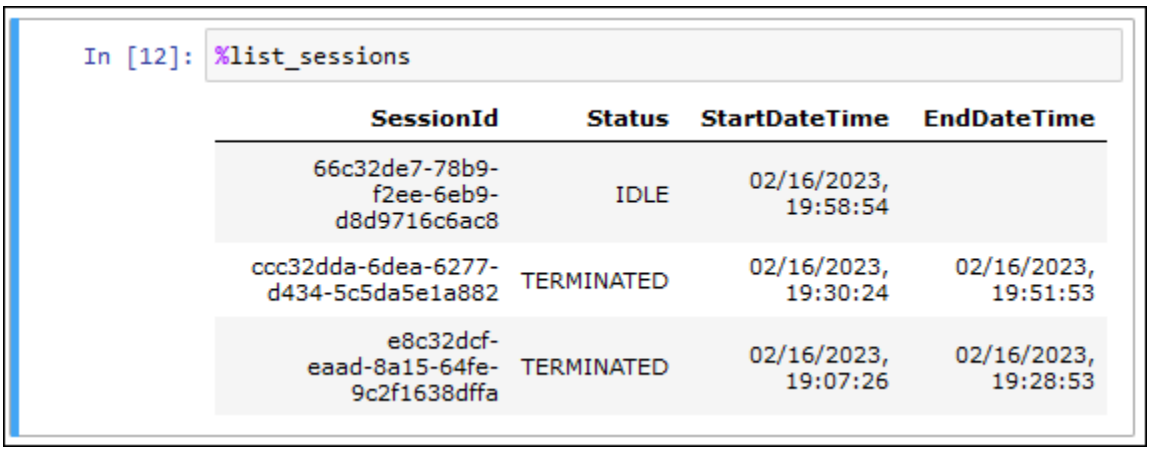
Affiche les descriptions des commandes magiques disponibles.



```
In [6]: %help

Available Magic Commands:
Magic | Input | Description
%session_id  | None | Return the session ID for the running session.
%status | None | Describes the current session and display SessionID, State,
WorkGroup, EngineVersion and StartTime
%help | None | Displays list of supported magics
%set_log_level | String | Sets the current log level to the provided log leve
ls (ERROR|INFO|WARNING etc)
%list_sessions | None | Lists the most recent sessions associated with the cu
rrent workgroup
%%sql | String | Run an SQL command against SparkSQL.
```

%list_sessions

Répertorie les sessions associées au bloc-notes. Les informations relatives à chaque session incluent l'ID de session, l'état de la session, ainsi que la date et l'heure de début et de fin de la session.

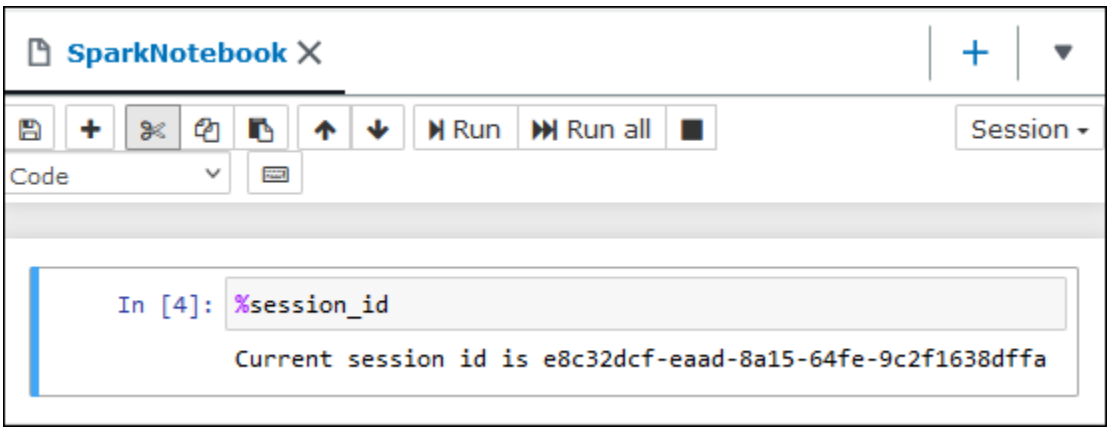


```
In [12]: %list_sessions
```

SessionId	Status	StartDateTime	EndDateTime
66c32de7-78b9-f2ee-6eb9-d8d9716c6ac8	IDLE	02/16/2023, 19:58:54	
ccc32dda-6dea-6277-d434-5c5da5e1a882	TERMINATED	02/16/2023, 19:30:24	02/16/2023, 19:51:53
e8c32dcf-eaad-8a15-64fe-9c2f1638dffa	TERMINATED	02/16/2023, 19:07:26	02/16/2023, 19:28:53

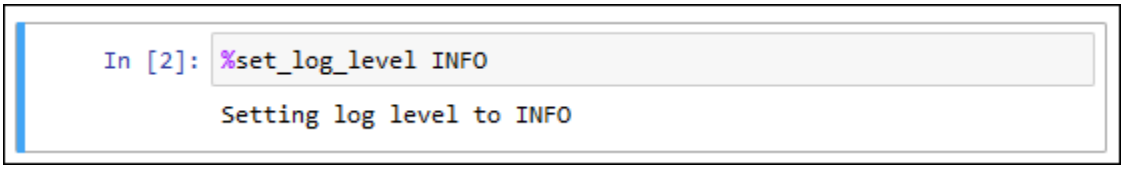
%session_id

Récupère l'ID de session en cours.



%set_log_level

Définit ou réinitialise l'enregistreur pour qu'il utilise le niveau de journalisation spécifié. Les valeurs admises sont `DEBUG`, `ERROR`, `FATAL`, `INFO` et `WARN` ou `WARNING`. Les valeurs doivent être en majuscules et ne doivent pas être entre guillemets simples ou doubles.



%status

Décrit la session en cours. La sortie inclut l'ID de session, l'état de la session, le nom du groupe de travail, la version du moteur PySpark et l'heure de début de session. Cette commande magique nécessite une session active pour récupérer les détails de la session.

Voici les valeurs d'état possibles :

EN COURS DE CRÉATION : la session est en cours de démarrage, y compris l'acquisition de ressources.

CRÉÉE : la session a été démarrée.

INACTIVE : la session est en mesure d'accepter un calcul.

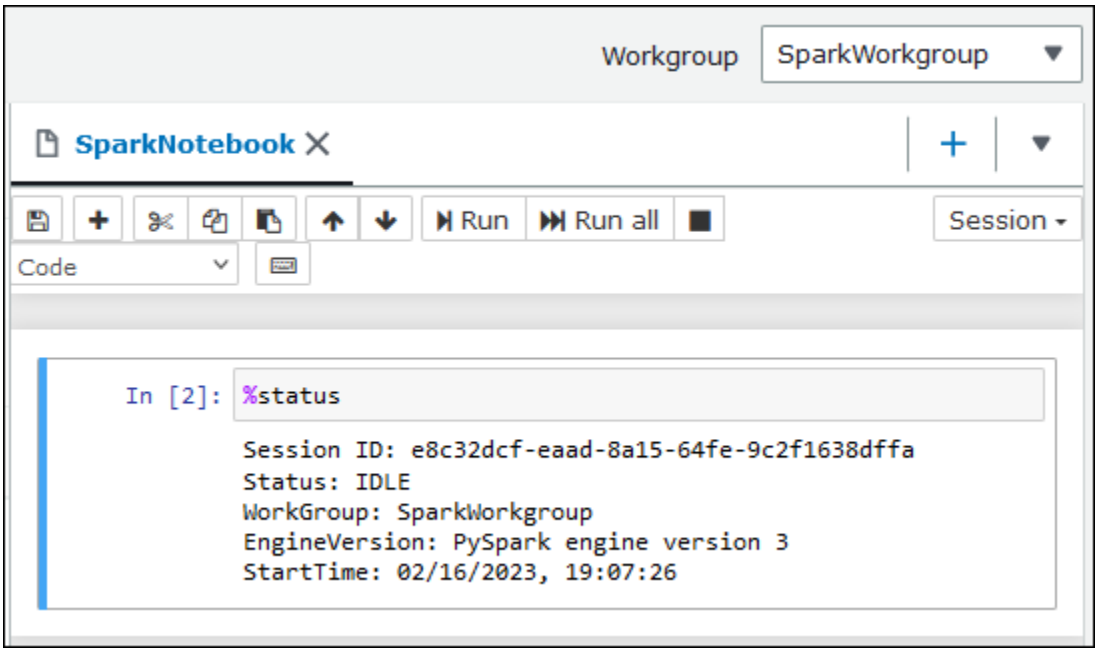
OCCUPÉE : La session est en train de traiter une autre tâche et n'est pas en mesure d'accepter un calcul.

EN COURS D'ARRÊT : la session est en cours d'arrêt.

ARRÊTÉE : la session et ses ressources ne sont plus en cours d'exécution.

DÉGRADÉE : la session ne compte aucun coordinateur sain.

ÉCHEC : en raison d'un échec, la session et ses ressources ne sont plus en cours d'exécution.



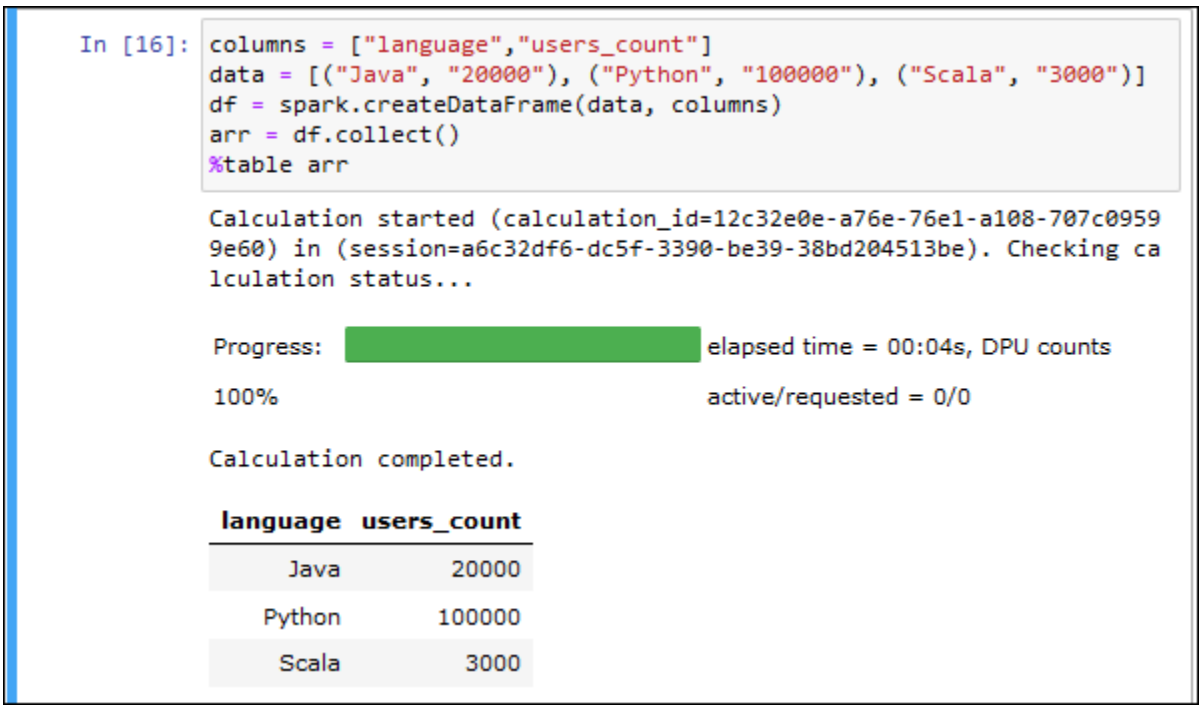
Magies pour créer des graphiques de données

Les magies linéraires présentées dans cette section sont spécialisés dans le rendu de données pour des types de données particuliers ou en association avec des bibliothèques graphiques.

%table

Vous pouvez utiliser la commande magique `%table` pour afficher les données du cadre de données sous forme de table.

L'exemple suivant crée un cadre de données comptant deux colonnes et trois lignes de données, puis affiche les données sous forme de table.



%matplot

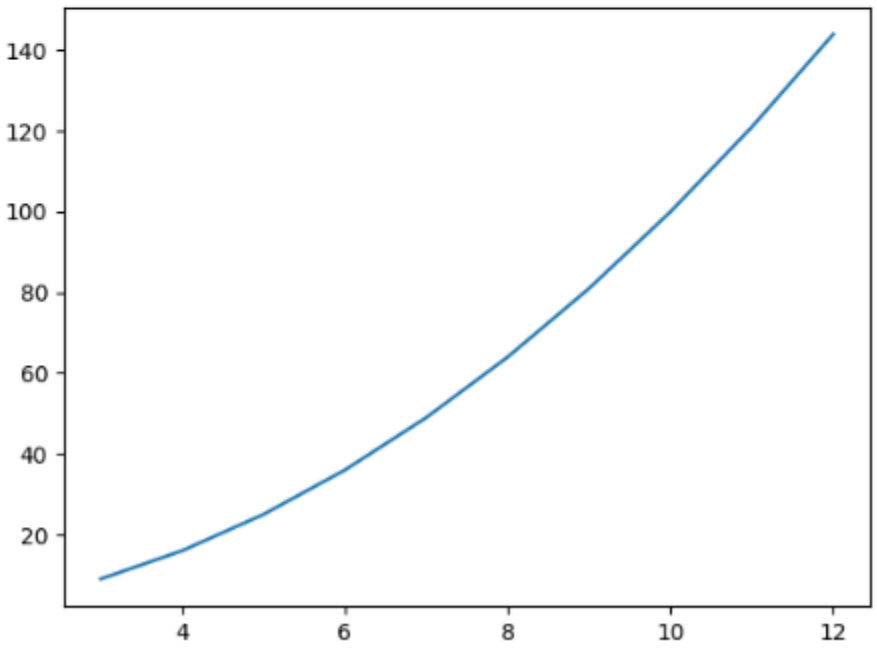
Matplotlib est une bibliothèque complète permettant de créer des visualisations statiques, animées et interactives dans Python. Vous pouvez utiliser la commande magique `%matplot` pour créer un graphique après avoir importé la bibliothèque matplotlib dans une cellule de bloc-notes.

L'exemple suivant importe la bibliothèque matplotlib, crée un ensemble de coordonnées x et y, puis utilise la commande magique `%matplot` pour créer un graphique des points.

```
import matplotlib.pyplot as plt
x=[3,4,5,6,7,8,9,10,11,12]
y= [9,16,25,36,49,64,81,100,121,144]
plt.plot(x,y)
%matplot plt
```

In [12]:
```
import matplotlib.pyplot as plt
x=[3,4,5,6,7,8,9,10,11,12]
y= [9,16,25,36,49,64,81,100,121,144]
plt.plot(x,y)
%matplot plt
```

```
Calculation started (calculation_id=5ac32e04-81b6-9ee7-ce55-539ee2ce383e) in (session=a6c32df6-dc5f-3390-be39-38bd204513be). Checking calculation status...

Progress:                                   elapsed time =
100%                                        00:02s

Calculation completed.
```




```
[<matplotlib.lines.Line2D object at 0x7f6e29e580>]
```

Utilisation conjointe des bibliothèques matplotlib et seaborn

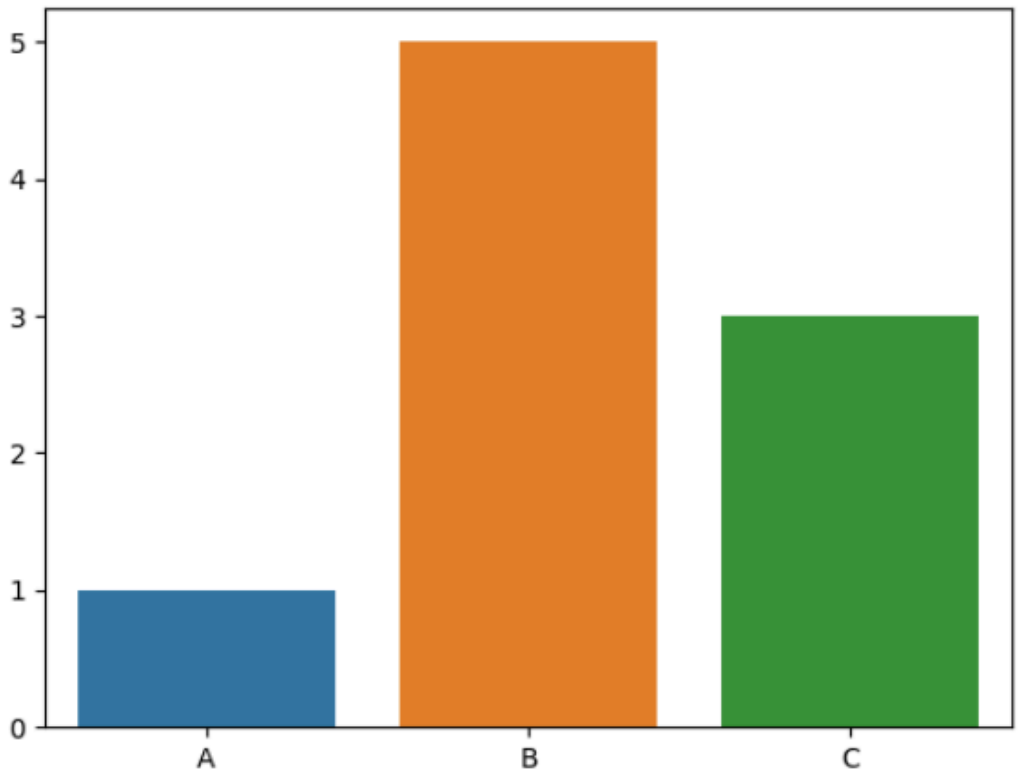
Seaborn est une bibliothèque permettant de créer des graphiques statistiques dans Python. Elle s'appuie sur matplotlib et s'intègre étroitement aux structures de données pandas (analyse de données Python). Vous pouvez également utiliser la commande magique `%matplot` pour afficher les données Seaborn.

L'exemple suivant utilise à la fois les bibliothèques matplotlib et seaborn pour créer un graphique à barres simple.

```
import matplotlib.pyplot as plt
import seaborn as sns

x = ['A', 'B', 'C']
y = [1, 5, 3]

sns.barplot(x, y)
%matplot plt
```

In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns


x = ['A', 'B', 'C']
y = [1, 5, 3]

sns.barplot(x, y)
%matplot plt
```

```
Calculation started (calculation_id=08c32e1b-233b-4a72-6571-1ae7a28a7b78)
in (session=64c32e1a-f45e-1d52-b54b-85202e2a9233). Checking calculation st
atus...

Progress: 100% elapsed time = 00:04s

Calculation completed.
```



%plotly

Plotly est une bibliothèque graphique open source pour Python que vous pouvez utiliser pour créer des graphiques interactifs. Vous utilisez la commande magique `%ploty` pour afficher les données Plotly.

L'exemple suivant utilise les bibliothèques StringIO, plotly et pandas sur les données sur les cours des actions pour créer un graphique de l'action entre février et mars 2015.

```
from io import StringIO
csvString = """
Date,AAPL.Open,AAPL.High,AAPL.Low,AAPL.Close,AAPL.Volume,AAPL.Adjusted,dn,mavg,up,direction
2015-02-17,127.489998,128.880005,126.919998,127.830002,63152400,122.905254,106.7410523,117.9276
2015-02-18,127.629997,128.779999,127.449997,128.720001,44891700,123.760965,107.842423,118.94033
2015-02-19,128.479996,129.029999,128.330002,128.449997,37362400,123.501363,108.8942449,119.8891
2015-02-20,128.619995,129.5,128.050003,129.5,48948400,124.510914,109.7854494,120.7635001,131.74
2015-02-23,130.020004,133,129.660004,133,70974100,127.876074,110.3725162,121.7201668,133.067817
2015-02-24,132.940002,133.600006,131.169998,132.169998,69228100,127.078049,111.0948689,122.6648
2015-02-25,131.559998,131.600006,128.149994,128.789993,74711700,123.828261,113.2119183,123.6296
2015-02-26,128.789993,130.869995,126.610001,130.419998,91287500,125.395469,114.1652991,124.2823
2015-02-27,130,130.570007,128.240005,128.460007,62014800,123.510987,114.9668484,124.8426669,134
2015-03-02,129.25,130.279999,128.300003,129.089996,48096700,124.116706,115.8770904,125.4036668,
2015-03-03,128.960007,129.520004,128.089996,129.360001,37816300,124.376308,116.9535132,125.9551
2015-03-04,129.100006,129.559998,128.320007,128.539993,31666300,123.587892,118.0874253,126.4730
2015-03-05,128.580002,128.75,125.760002,126.410004,56517100,121.539962,119.1048311,126.848667,1
2015-03-06,128.399994,129.369995,126.260002,126.599998,72842100,121.722637,120.190797,127.22883
2015-03-09,127.959999,129.570007,125.059998,127.139999,88528500,122.241834,121.6289771,127.6311
2015-03-10,126.410004,127.220001,123.800003,124.510002,68856600,119.71316,123.1164763,127.92350
"""
csvStringIO = StringIO(csvString)

from io import StringIO
import plotly.graph_objects as go
import pandas as pd
from datetime import datetime
df = pd.read_csv(csvStringIO)
fig = go.Figure(data=[go.Candlestick(x=df['Date'],
open=df['AAPL.Open'],
high=df['AAPL.High'],
low=df['AAPL.Low'],
close=df['AAPL.Close'])])
%plotly fig
```

Gestion des fichiers de bloc-notes

Outre l'utilisation de l'explorateur de blocs-notes pour créer et ouvrir des blocs-notes, vous pouvez également l'utiliser pour renommer, supprimer, exporter ou importer des blocs-notes, ou encore consulter l'historique des sessions d'un bloc-notes.

Renommer un bloc-notes

1. Terminez toutes les sessions actives du bloc-notes que vous voulez renommer. Les sessions actives du bloc-notes doivent être terminées pour pouvoir le renommer.
2. Ouvrez Notebook explorer (Explorateur de bloc-notes).

3. Dans la liste des blocs-notes, sélectionnez le bouton d'option correspondant au bloc-notes que vous souhaitez renommer.
4. Dans le menu Actions, sélectionnez Rename (Renommer).
5. À l'invite Rename notebook (Renommer le bloc-notes), saisissez le nouveau nom, puis sélectionnez Save (Enregistrer). Le nouveau nom du bloc-notes apparaît dans la liste des blocs-notes.

Supprimer un bloc-notes

1. Terminez toutes les sessions actives du bloc-notes que vous voulez supprimer. Les sessions actives du bloc-notes doivent être terminées pour pouvoir le supprimer.
2. Ouvrez Notebook explorer (Explorateur de bloc-notes).
3. Dans la liste des blocs-notes, sélectionnez le bouton d'option correspondant au bloc-notes que vous souhaitez supprimer.
4. Dans le menu Actions, choisissez Delete (Supprimer).
5. À l'invite Delete notebook? (Supprimer le bloc-notes ?), saisissez le nom du bloc-notes, puis choisissez Delete (Supprimer) pour confirmer la suppression. Le nom du bloc-notes est supprimé de la liste des blocs-notes.

Exporter un bloc-notes

1. Ouvrez Notebook explorer (Explorateur de bloc-notes).
2. Dans la liste des blocs-notes, sélectionnez le bouton d'option correspondant au bloc-notes que vous souhaitez exporter.
3. Dans le menu Actions, sélectionnez Export file (Exporter un fichier).

Importer un bloc-notes

1. Ouvrez Notebook explorer (Explorateur de bloc-notes).
2. Choisissez Import file (Importer un fichier).
3. Accédez à l'emplacement du fichier que vous voulez importer sur votre ordinateur local, puis choisissez Open (Ouvrir). Le bloc-notes importé apparaît dans la liste des blocs-notes.

Consulter l'historique des sessions d'un bloc-notes

1. Ouvrez Notebook explorer (Explorateur de bloc-notes).
2. Dans la liste des blocs-notes, sélectionnez le bouton d'option du bloc-notes dont vous voulez consulter l'historique des sessions.
3. Dans le menu Actions, choisissez Session history (Historique des sessions).
4. Dans l'onglet History (Historique), choisissez un Session ID (ID de session) pour en consulter les informations sur la session et ses calculs.

Utilisation de formats de table autres que Hive dans Amazon Athena pour Apache Spark

Lorsque vous travaillez avec des sessions et des blocs-notes dans Athena pour Spark, vous pouvez utiliser les tables Linux Foundation Delta Lake, Apache Hudi et Apache Iceberg, en plus des tables Apache Hive.

Considérations et restrictions

Lorsque vous utilisez des formats de table autres qu'Apache Hive avec Athena pour Spark, tenez compte des points suivants :

- Outre Apache Hive, un seul format de table est pris en charge par bloc-notes. Pour utiliser plusieurs formats de table dans Athena pour Spark, créez un bloc-notes distinct pour chaque format de table. Pour plus d'informations sur la création de blocs-notes dans Athena pour Spark, consultez Création de votre propre bloc-notes.
- Les formats de table Delta Lake, Hudi et Iceberg ont été testés sur Athena pour Spark en les utilisant AWS Glue comme métastore. Vous pouvez peut-être utiliser d'autres métastores, mais cette utilisation n'est actuellement pas prise en charge.
- Pour utiliser les formats de table supplémentaires, remplacez la propriété `spark_catalog` par défaut, comme indiqué dans la console Athena et dans cette documentation. Ces catalogues autres que Hive peuvent lire les tables Hive, en plus de leurs propres formats de table.

Versions de table

Le tableau suivant reprend les versions de table autres que Hive prises en charge dans Amazon Athena pour Apache Spark.

Format de table	Version prise en charge
Apache Iceberg	1.2.1
Apache Hudi	0,13
Linux Foundation Delta Lake	2.0.2

Dans Athena pour Spark, ces fichiers `.jar` de format de table et leurs dépendances sont chargés dans le chemin de classe des pilotes et exécuteurs Spark.

Pour consulter un article de blog sur le AWS Big Data expliquant comment utiliser les formats de table Iceberg, Hudi et Delta Lake à l'aide de Spark SQL dans les blocs-notes Amazon Athena, consultez Utiliser Amazon Athena avec Spark SQL pour vos formats de tables transactionnels open source.

Rubriques

- Apache Iceberg
- Apache Hudi
- Linux Foundation Delta Lake

Apache Iceberg

Apache Iceberg est un format de table ouvert pour les jeux de données volumineux dans Amazon Simple Storage Service (Amazon S3). Il vous fournit des performances de requête rapides sur de grandes tables, des validations atomiques, des écritures simultanées et une évolution de table compatible avec SQL.

Pour utiliser les tables Apache Iceberg dans Athena pour Spark, configurez les propriétés Spark suivantes. Ces propriétés sont configurées pour vous par défaut dans la console Athena pour Spark lorsque vous choisissez Apache Iceberg comme format de table. Pour les étapes, consultez Modification des détails de la session ou Création de votre propre bloc-notes.

```
"spark.sql.catalog.spark_catalog": "org.apache.iceberg.spark.SparkSessionCatalog",
"spark.sql.catalog.spark_catalog.catalog-impl":
 "org.apache.iceberg.aws.glue.GlueCatalog",
"spark.sql.catalog.spark_catalog.io-impl": "org.apache.iceberg.aws.s3.S3FileIO",
```

```
"spark.sql.extensions":
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
```

La procédure suivante explique comment utiliser une table Apache Iceberg dans un bloc-notes Athena pour Spark. Exécutez chaque étape dans une nouvelle cellule du bloc-notes.

Pour utiliser une table Apache Iceberg dans Athena pour Spark.

1. Définissez les constantes à utiliser dans le bloc-notes.

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
```

2. Créez un Apache Spark DataFrame.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Créez une base de données.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Créez une table Apache Iceberg vide.

```
spark.sql("""
CREATE TABLE {}.{} (
language string,
users_count int
) USING ICEBERG
""".format(DB_NAME, TABLE_NAME))
```

5. Insérez une ligne de données dans la table.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME, TABLE_NAME))
```

6. Confirmez que vous pouvez interroger la nouvelle table.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Pour plus d'informations et des exemples sur l'utilisation des tables Spark DataFrames et Iceberg, consultez la section Spark Queries dans la documentation d'Apache Iceberg.

Apache Hudi

Apache Hudi est un cadre de gestion de données open source qui simplifie le traitement progressif des données. Les actions d'insertion, de mise à jour, de mise à jour/insertion et de suppression au niveau de l'enregistrement sont traitées avec une plus grande précision, ce qui réduit la surcharge.

Pour utiliser les tables Apache Hudi dans Athena pour Spark, configurez les propriétés Spark suivantes. Ces propriétés sont configurées pour vous par défaut dans la console Athena pour Spark lorsque vous choisissez Apache Hudi comme format de table. Pour les étapes, consultez Modification des détails de la session ou Création de votre propre bloc-notes.

```
"spark.sql.catalog.spark_catalog": "org.apache.spark.sql.hudi.catalog.HoodieCatalog",
"spark.serializer": "org.apache.spark.serializer.KryoSerializer",
"spark.sql.extensions": "org.apache.spark.sql.hudi.HoodieSparkSessionExtension"
```

La procédure suivante explique comment utiliser une table Apache Hudi dans un bloc-notes Athena pour Spark. Exécutez chaque étape dans une nouvelle cellule du bloc-notes.

Pour utiliser une table Apache Hudi dans Athena pour Spark.

1. Définissez les constantes à utiliser dans le bloc-notes.

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
```

2. Créez un Apache Spark DataFrame.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Créez une base de données.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Créez une table Apache Hudi vide.

```
spark.sql("""
CREATE TABLE {}.{} (
language string,
users_count int
) USING HUDI
TBLPROPERTIES (
primaryKey = 'language',
type = 'mor'
);
""".format(DB_NAME, TABLE_NAME))
```

5. Insérez une ligne de données dans la table.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME,TABLE_NAME))
```

6. Confirmez que vous pouvez interroger la nouvelle table.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Linux Foundation Delta Lake

Linux Foundation Delta Lake est un format de table que vous pouvez utiliser pour l'analytique du big data. Vous pouvez utiliser Athena pour Spark pour lire directement les tables de Delta Lake stockées dans Amazon S3.

Pour utiliser les tables Delta Lake dans Athena pour Spark, configurez les propriétés Spark suivantes. Ces propriétés sont configurées pour vous par défaut dans la console Athena pour Spark lorsque vous choisissez Delta Lake comme format de table. Pour les étapes, consultez Modification des détails de la session ou Création de votre propre bloc-notes.

```
"spark.sql.catalog.spark_catalog" : "org.apache.spark.sql.delta.catalog.DeltaCatalog",
"spark.sql.extensions" : "io.delta.sql.DeltaSparkSessionExtension"
```

La procédure suivante explique comment utiliser une table Delta Lake dans un bloc-notes Athena pour Spark. Exécutez chaque étape dans une nouvelle cellule du bloc-notes.

Pour utiliser une table Delta Lake dans Athena pour Spark

1. Définissez les constantes à utiliser dans le bloc-notes.

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
```

2. Créez un Apache Spark DataFrame.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Créez une base de données.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Créez une table Delta Lake vide.

```
spark.sql("""
CREATE TABLE {}.{} (
  language string,
  users_count int
) USING DELTA
""".format(DB_NAME, TABLE_NAME))
```

5. Insérez une ligne de données dans la table.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME, TABLE_NAME))
```

6. Confirmez que vous pouvez interroger la nouvelle table.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Prise en charge de la bibliothèque Python dans Amazon Athena pour Apache Spark

Cette page décrit la terminologie utilisée et la gestion du cycle de vie suivie pour les moteurs d'exécution, les bibliothèques et les packages utilisés dans Amazon Athena pour Apache Spark.

Définitions

- Amazon Athena pour Apache Spark est une version personnalisée d'Apache Spark open source. Pour voir la version actuelle, exécutez la commande `print(f'{spark.version}')` dans une cellule du bloc-notes.
- Le moteur d'exécution Athena est l'environnement dans lequel votre code s'exécute. L'environnement inclut un interpréteur Python et PySpark des bibliothèques.
- La bibliothèque ou le package externe est un fichier JAR Java ou Scala ou une bibliothèque Python qui ne fait pas partie du moteur d'exécution Athena, mais qui peut être inclus dans les tâches Athena pour Spark. Les packages externes peuvent être créés par Amazon ou par vous.
- Le package pratique est un ensemble de packages externes sélectionnés par Athena que vous pouvez choisir d'inclure dans vos applications Spark.
- L'offre groupée combine le moteur d'exécution Athena et un package pratique.
- La bibliothèque utilisateur est une bibliothèque externe ou un package que vous ajoutez explicitement à votre tâche Athena pour Spark.
  - La bibliothèque utilisateur est un package externe qui ne fait pas partie d'un package pratique. La bibliothèque utilisateur nécessite un chargement et une installation, comme lorsque vous écrivez des fichiers `.py`, que vous les compressez et que vous ajoutez le fichier `.zip` à votre application.
- L'application Athena pour Spark est une tâche ou une requête que vous soumettez à Athena pour Spark.

Gestion des cycles de vie

Gestion des versions et obsolescence du moteur d'exécution

Le composant principal du moteur d'exécution Athena est l'interpréteur Python. Python étant un langage évolutif, de nouvelles versions sont publiées régulièrement et la prise en charge des anciennes versions est supprimée. Athena ne vous recommande pas d'exécuter des programmes

avec des versions dépréciées de l'interpréteur Python et vous recommande vivement d'utiliser la dernière version du moteur d'exécution Athena chaque fois que cela est possible.

Le calendrier d'obsolescence du moteur d'exécution d'Athena est le suivant :

1. Après qu'Athena ait fourni un nouveau moteur d'exécution, Athena continuera à prendre en charge le moteur d'exécution précédent pendant 6 mois. Pendant cette période, Athena appliquera des correctifs de sécurité et des mises à jour au moteur d'exécution précédent.
2. Après 6 mois, Athena mettra fin à la prise en charge de la version précédente. Athena n'appliquera plus les correctifs de sécurité et autres mises à jour du moteur d'exécution précédent. Les applications Spark utilisant l'ancien moteur d'exécution ne pourront plus bénéficier du support technique.
3. Après 12 mois, vous ne pourrez plus mettre à jour ou modifier les applications Spark dans un groupe de travail qui utilise le moteur d'exécution précédent. Nous vous recommandons de mettre à jour vos applications Spark avant la fin de cette période. Après la fin de la période, vous pouvez toujours exécuter les blocs-notes existants, mais tous les blocs-notes qui utilisent encore le moteur d'exécution précédent recevront un avertissement à cet effet.
4. Après 18 mois, vous ne pourrez plus exécuter de tâches dans le groupe de travail en utilisant le moteur d'exécution précédent.

Gestion des versions et obsolescence des packages pratiques

Le contenu des packages pratiques évolue au fil du temps. Athena ajoute, supprime ou améliore occasionnellement ces packages pratiques.

Athena applique les directives suivantes pour les packages pratiques :

- Les packages pratiques ont un schéma de gestion des versions simple comme 1, 2, 3.
- Chaque version de package pratique comprend des versions spécifiques de packages externes. Une fois qu'Athena a créé un package pratique, l'ensemble des packages externes du package pratique et leurs versions correspondantes ne changent pas.
- Athena crée une nouvelle version de package pratique lorsqu'elle inclut un nouveau package externe, supprime un package externe ou met à niveau la version d'un ou de plusieurs packages externes.

Athena rend obsolète un package pratique lorsqu'elle rend obsolète le moteur d'exécution Athena que le package utilise. Athena peut rendre les paquets obsolètes plus tôt afin de limiter le nombre de packages qu'elle prend en charge.

Le calendrier d'obsolescence des packages pratiques suit le calendrier d'obsolescence du moteur d'exécution d'Athena.

Liste des bibliothèques Python préinstallées

Les bibliothèques Python préinstallées comprennent les bibliothèques suivantes.

```
boto3==1.24.31
botocore==1.27.31
certifi==2022.6.15
charset-normalizer==2.1.0
cycler==0.11.0
cython==0.29.30
docutils==0.19
fonttools==4.34.4
idna==3.3
jmespath==1.0.1
joblib==1.1.0
kiwisolver==1.4.4
matplotlib==3.5.2
mpmath==1.2.1
numpy==1.23.1
packaging==21.3
pandas==1.4.3
patsy==0.5.2
pillow==9.2.0
plotly==5.9.0
pmdarima==1.8.5
pyathena==2.9.6
pyparsing==3.0.9
python-dateutil==2.8.2
pytz==2022.1
requests==2.28.1
s3transfer==0.6.0
scikit-learn==1.1.1
scipy==1.8.1
seaborn==0.11.2
six==1.16.0
statsmodels==0.13.2
```

```
sympy==1.10.1
tenacity==8.0.1
threadpoolctl==3.1.0
urllib3==1.26.10
pyarrow==9.0.0
```

Remarques

- MLLib (bibliothèque d'apprentissage automatique Apache Spark) et le `pyspark.ml` package ne sont pas pris en charge.
- Actuellement, n'`pip install`est pas pris en charge dans les sessions Athena pour Spark.

Pour plus d'informations sur l'importation de bibliothèques Python dans Amazon Athena pour Apache Spark, consultez. Importation de fichiers et de bibliothèques Python dans Amazon Athena pour Apache Spark

Importation de fichiers et de bibliothèques Python dans Amazon Athena pour Apache Spark

Ce document fournit des exemples sur la manière d'importer des fichiers et des bibliothèques Python vers Amazon Athena pour Apache Spark.

Considérations et restrictions

- Version Python : Athena pour Spark utilise actuellement la version 3.9.16 de Python. Notez que les packages Python sont sensibles aux versions mineures de Python.
- Athena pour architecture Spark : Athena pour Spark utilise Amazon Linux 2 sur l'architecture ARM64. Notez que certaines bibliothèques Python ne distribuent pas de fichiers binaires pour cette architecture.
- Objets partagés binaires (SO) : comme la SparkContext addPyFileméthode ne détecte pas les objets partagés binaires, elle ne peut pas être utilisée dans Athena for Spark pour ajouter des packages Python qui dépendent d'objets partagés.
- Jeux de données distribués résilients (RDD) : les RDD ne sont pas pris en charge.
- DataFrame.forEach — La méthode .foreach n'est pas prise en PySpark DataFramecharge.

Exemples

Les exemples utilisent les conventions suivantes.

- Emplacement Amazon S3 réservé `s3://DOC-EXAMPLE-BUCKET`. Remplacez-le par l'emplacement de votre propre compartiment S3.
- Tous les blocs de code qui s'exécutent à partir d'un shell Unix sont affichés sous le nom de *`directory_name`* $. Par exemple, la commande `ls` dans le répertoire `/tmp` et sa sortie s'affichent comme suit :

  ```
  /tmp $ ls
  ```

  Sortie

  ```
  file1 file2
  ```

- Ajout d'un fichier à un bloc-notes après son écriture dans le répertoire temporaire local
- Importation d'un fichier à partir d'Amazon S3
- Ajout de fichiers Python et enregistrement d'un UDF
- Importation d'un fichier .zip Python
- Importation de deux versions d'une bibliothèque Python en tant que modules distincts
- Importation d'un fichier .zip Python à partir de PyPI
- Importation d'un fichier .zip Python comportant des dépendances à partir de PyPI

Importation de fichiers texte à utiliser dans les calculs

Les exemples de cette rubrique montrent comment importer des fichiers texte pour les utiliser dans les calculs de vos carnets dans Athena pour Spark.

Ajout d'un fichier à un bloc-notes après son écriture dans le répertoire temporaire local

L'exemple suivant montre comment écrire un fichier dans un répertoire temporaire local, l'ajouter à un bloc-notes et le tester.

```
import os
from pyspark import SparkFiles
tempdir = '/tmp/'
```

```
path = os.path.join(tempdir, "test.txt")
with open(path, "w") as testFile:
    _ = testFile.write("5")
sc.addFile(path)

def func(iterator):
    with open(SparkFiles.get("test.txt")) as testFile:
        fileVal = int(testFile.readline())
        return [x * fileVal for x in iterator]

#Test the file
from pyspark.sql.functions import udf
from pyspark.sql.functions import col

udf_with_import = udf(func)
df = spark.createDataFrame([(1, "a"), (2, "b")])
df.withColumn("col", udf_with_import(col('_2'))).show()
```

Sortie

```
Calculation completed.
+---+---+-------+
| _1| _2|    col|
+---+---+-------+
|  1|  a|[aaaaa]|
|  2|  b|[bbbbb]|
+---+---+-------+
```

Importation d'un fichier à partir d'Amazon S3

L'exemple suivant montre comment importer un fichier à partir d'Amazon S3 dans un bloc-notes et le tester.

Importation d'un fichier à partir d'Amazon S3 dans un bloc-notes

1. Créez un fichier `test.txt` dont le nom comporte une seule ligne contenant la valeur `5`.
2. Ajoutez le fichier à un compartiment dans Amazon S3. Cet exemple utilise l'emplacement `s3://DOC-EXAMPLE-BUCKET`.
3. Utilisez le code suivant pour importer le fichier dans votre bloc-notes et le tester.

```
from pyspark import SparkFiles
sc.addFile('s3://DOC-EXAMPLE-BUCKET/test.txt')
```

```
def func(iterator):
   with open(SparkFiles.get("test.txt")) as testFile:
       fileVal = int(testFile.readline())
       return [x * fileVal for x in iterator]

#Test the file
from pyspark.sql.functions import udf
from pyspark.sql.functions import col

udf_with_import = udf(func)
df = spark.createDataFrame([(1, "a"), (2, "b")])
df.withColumn("col", udf_with_import(col('_2'))).show()
```

Sortie

```
Calculation completed.
+---+---+-------+
| _1| _2|    col|
+---+---+-------+
|  1|  a|[aaaaa]|
|  2|  b|[bbbbb]|
+---+---+-------+
```

Ajout de fichiers Python

Les exemples de cette rubrique montrent comment ajouter des fichiers et des bibliothèques Python à vos blocs-notes Spark dans Athena.

Ajout de fichiers Python et enregistrement d'un UDF

L'exemple suivant montre comment ajouter des fichiers Python à partir d'Amazon S3 à votre bloc-notes et enregistrer un UDF.

Ajout de fichiers Python à votre bloc-notes et enregistrement d'un UDF

1. En utilisant votre propre emplacement Amazon S3, créez le fichier `s3://DOC-EXAMPLE-BUCKET/file1.py` avec le contenu suivant :

```
def xyz(input):
    return 'xyz  - udf ' + str(input);
```

2. Dans le même emplacement S3, créez le fichier `s3://DOC-EXAMPLE-BUCKET/file2.py` avec le contenu suivant :

```
from file1 import xyz
def uvw(input):
    return 'uvw -> ' + xyz(input);
```

3. Dans votre bloc-notes Athena pour Spark, exécutez les commandes suivantes.

```
sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/file1.py')
sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/file2.py')

def func(iterator):
    from file2 import uvw
    return [uvw(x) for x in iterator]

from pyspark.sql.functions import udf
from pyspark.sql.functions import col

udf_with_import = udf(func)

df = spark.createDataFrame([(1, "a"), (2, "b")])

df.withColumn("col", udf_with_import(col('_2'))).show(10)
```

Sortie

```
Calculation started (calculation_id=1ec09e01-3dec-a096-00ea-57289cdb8ce7) in
 (session=c8c09e00-6f20-41e5-98bd-4024913d6cee). Checking calculation status...
Calculation completed.
+---+---+--------------------+
| _1| _2|                 col|
+---+---+--------------------+
| 1 |  a|[uvw -> xyz - ud... |
| 2 |  b|[uvw -> xyz - ud... |
+---+---+--------------------+
```

Importation d'un fichier .zip Python

Vous pouvez utiliser Python `addPyFile` et ses méthodes `import` pour importer un fichier .zip Python dans votre bloc-notes.

> **Note**
>
> Les fichiers `.zip` que vous importez dans Athena Spark peuvent inclure uniquement des packages Python. Par exemple, l'inclusion de packages contenant des fichiers basés sur le langage C n'est pas prise en charge.

Pour importer un fichier **`.zip`** Python dans votre bloc-notes

1. Sur votre ordinateur local, dans un répertoire de bureau tel que `\tmp`, créez un répertoire appelé `moduletest`.
2. Dans le répertoire `moduletest`, créez un fichier nommé `hello.py` avec les contenus suivants :

   ```
   def hi(input):
       return 'hi ' + str(input);
   ```

3. Dans le même répertoire, ajoutez un fichier vide portant le nom `__init__.py`.

   Si vous listez le contenu du répertoire, il devrait maintenant ressembler à ce qui suit.

   ```
   /tmp $ ls moduletest
   __init__.py       hello.py
   ```

4. Utilisez la commande `zip` pour placer les deux fichiers du module dans un fichier appelé `moduletest.zip`.

   ```
   moduletest $ zip -r9 ../moduletest.zip *
   ```

5. Chargez les fichiers `.zip` dans votre compartiment Amazon S3.
6. Utilisez le code suivant pour importer le fichier `.zip` Python dans votre bloc-notes.

   ```
   sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/moduletest.zip')

   from moduletest.hello import hi

   from pyspark.sql.functions import udf
   from pyspark.sql.functions import col

   hi_udf = udf(hi)
   ```

```
df = spark.createDataFrame([(1, "a"), (2, "b")])

df.withColumn("col", hi_udf(col('_2'))).show()
```

Sortie

```
Calculation started (calculation_id=6ec09e8c-6fe0-4547-5f1b-6b01adb2242c) in
 (session=dcc09e8c-3f80-9cdc-bfc5-7effa1686b76). Checking calculation status...
Calculation completed.
+---+---+----+
| _1| _2| col|
+---+---+----+
|  1|  a|hi a|
|  2|  b|hi b|
+---+---+----+
```

Importation de deux versions d'une bibliothèque Python en tant que modules distincts

Les exemples de code suivants montrent comment ajouter et importer deux versions différentes d'une bibliothèque Python à partir d'un emplacement dans Amazon S3 en tant que deux modules distincts. Le code ajoute chaque fichier de la bibliothèque à partir de S3, l'importe, puis imprime la version de la bibliothèque pour vérifier l'importation.

```
sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/python-third-party-libs-test/
simplejson_v3_15.zip')
sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/python-third-party-libs-test/
simplejson_v3_17_6.zip')

import simplejson_v3_15
print(simplejson_v3_15.__version__)
```

Sortie

```
3.15.0
```

```
import simplejson_v3_17_6
print(simplejson_v3_17_6.__version__)
```

Sortie

```
3.17.6
```

Importation d'un fichier .zip Python à partir de PyPI

Cet exemple utilise la commande `pip` pour télécharger un fichier .zip Python du projet bpabel/piglatin à partir du référentiel Python Package Index (PyPI).

Importer un fichier .zip Python à partir de PyPI

1. Sur votre bureau local, utilisez les commandes suivantes pour créer un répertoire appelé `testpiglatin` et un environnement virtuel.

   ```
   /tmp $ mkdir testpiglatin
   /tmp $ cd testpiglatin
   testpiglatin $ virtualenv .
   ```

   Sortie

   ```
   created virtual environment CPython3.9.6.final.0-64 in 410ms
   creator CPython3Posix(dest=/private/tmp/testpiglatin, clear=False,
    no_vcs_ignore=False, global=False)
   seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle,
    via=copy, app_data_dir=/Users/user1/Library/Application Support/virtualenv)
   added seed packages: pip==22.0.4, setuptools==62.1.0, wheel==0.37.1
   activators
    BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonAct
   ```

2. Créez un sous-répertoire appelé `unpacked` pour contenir le projet.

   ```
   testpiglatin $ mkdir unpacked
   ```

3. Utilisez la commande `pip` pour installer le projet dans le répertoire `unpacked`.

   ```
   testpiglatin $ bin/pip install -t $PWD/unpacked piglatin
   ```

   Sortie

   ```
   Collecting piglatin
   Using cached piglatin-1.0.6-py2.py3-none-any.whl (3.1 kB)
   Installing collected packages: piglatin
   ```

```
Successfully installed piglatin-1.0.6
```

4. Vérifiez le contenu du répertoire.

```
testpiglatin $ ls
```

Sortie

```
bin lib pyvenv.cfg unpacked
```

5. Accédez au répertoire `unpacked` et affichez le contenu.

```
testpiglatin $ cd unpacked
unpacked $ ls
```

Sortie

```
piglatin piglatin-1.0.6.dist-info
```

6. Utilisez la commande `zip` pour placer le contenu du projet piglatin dans un fichier appelé `library.zip`.

```
unpacked $ zip -r9 ../library.zip *
```

Sortie

```
adding: piglatin/ (stored 0%)
adding: piglatin/__init__.py (deflated 56%)
adding: piglatin/__pycache__/ (stored 0%)
adding: piglatin/__pycache__/__init__.cpython-39.pyc (deflated 31%)
adding: piglatin-1.0.6.dist-info/ (stored 0%)
adding: piglatin-1.0.6.dist-info/RECORD (deflated 39%)
adding: piglatin-1.0.6.dist-info/LICENSE (deflated 41%)
adding: piglatin-1.0.6.dist-info/WHEEL (deflated 15%)
adding: piglatin-1.0.6.dist-info/REQUESTED (stored 0%)
adding: piglatin-1.0.6.dist-info/INSTALLER (stored 0%)
adding: piglatin-1.0.6.dist-info/METADATA (deflated 48%)
```

7. (Facultatif) Utilisez les commandes suivantes pour tester l'importation localement.

a. Définissez le chemin Python vers l'emplacement du fichier `library.zip` et démarrez Python.

```
/home $ PYTHONPATH=/tmp/testpiglatin/library.zip
/home $ python3
```

Sortie

```
Python 3.9.6 (default, Jun 29 2021, 06:20:32)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
```

b. Importez la bibliothèque et exécutez une commande de test.

```
>>> import piglatin
>>> piglatin.translate('hello')
```

Sortie

```
'ello-hay'
```

8. Utilisez des commandes telles que les suivantes pour ajouter le fichier `.zip` à partir d'Amazon S3, l'importer dans votre bloc-notes dans Athena et le tester.

```
sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/library.zip')

import piglatin
piglatin.translate('hello')

from pyspark.sql.functions import udf
from pyspark.sql.functions import col

hi_udf = udf(piglatin.translate)

df = spark.createDataFrame([(1, "hello"), (2, "world")])

df.withColumn("col", hi_udf(col('_2'))).show()
```

Sortie

```
Calculation started (calculation_id=e2c0a06e-f45d-d96d-9b8c-ff6a58b2a525) in
 (session=82c0a06d-d60e-8c66-5d12-23bcd55a6457). Checking calculation status...
Calculation completed.
+---+-----+--------+
| _1|   _2|     col|
+---+-----+--------+
|  1|hello|ello-hay|
|  2|world|orld-way|
+---+-----+--------+
```

Importation d'un fichier .zip Python comportant des dépendances à partir de PyPI

Cet exemple importe le package md2gemini, qui convertit le texte en Markdown au format de texte Gemini, à partir de PyPI. Le package a les dépendances suivantes :

```
cjkwrap
mistune
wcwidth
```

Importation d'un fichier .zip Python comportant des dépendances

1. Sur votre ordinateur local, utilisez les commandes suivantes pour créer un répertoire appelé `testmd2gemini` et un environnement virtuel.

   ```
   /tmp $ mkdir testmd2gemini
   /tmp $ cd testmd2gemini
   testmd2gemini$ virtualenv .
   ```

2. Créez un sous-répertoire appelé `unpacked` pour contenir le projet.

   ```
   testmd2gemini $ mkdir unpacked
   ```

3. Utilisez la commande `pip` pour installer le projet dans le répertoire `unpacked`.

   ```
   /testmd2gemini $ bin/pip install -t $PWD/unpacked md2gemini
   ```

   Sortie

   ```
   Collecting md2gemini
   ```

```
  Downloading md2gemini-1.9.0-py3-none-any.whl (31 kB)
Collecting wcwidth
  Downloading wcwidth-0.2.5-py2.py3-none-any.whl (30 kB)
Collecting mistune<3,>=2.0.0
  Downloading mistune-2.0.2-py2.py3-none-any.whl (24 kB)
Collecting cjkwrap
  Downloading CJKwrap-2.2-py2.py3-none-any.whl (4.3 kB)
Installing collected packages: wcwidth, mistune, cjkwrap, md2gemini
Successfully installed cjkwrap-2.2 md2gemini-1.9.0 mistune-2.0.2 wcwidth-0.2.5
...
```

4. Accédez au répertoire `unpacked` et vérifiez le contenu.

```
testmd2gemini $ cd unpacked
unpacked $ ls -lah
```

Sortie

```
total 16
drwxr-xr-x  13 user1  wheel   416B Jun  7 18:43 .
drwxr-xr-x   8 user1  wheel   256B Jun  7 18:44 ..
drwxr-xr-x   9 user1  staff   288B Jun  7 18:43 CJKwrap-2.2.dist-info
drwxr-xr-x   3 user1  staff    96B Jun  7 18:43 __pycache__
drwxr-xr-x   3 user1  staff    96B Jun  7 18:43 bin
-rw-r--r--   1 user1  staff   5.0K Jun  7 18:43 cjkwrap.py
drwxr-xr-x   7 user1  staff   224B Jun  7 18:43 md2gemini
drwxr-xr-x  10 user1  staff   320B Jun  7 18:43 md2gemini-1.9.0.dist-info
drwxr-xr-x  12 user1  staff   384B Jun  7 18:43 mistune
drwxr-xr-x   8 user1  staff   256B Jun  7 18:43 mistune-2.0.2.dist-info
drwxr-xr-x  16 user1  staff   512B Jun  7 18:43 tests
drwxr-xr-x  10 user1  staff   320B Jun  7 18:43 wcwidth
drwxr-xr-x   9 user1  staff   288B Jun  7 18:43 wcwidth-0.2.5.dist-info
```

5. Utilisez la commande `zip` pour placer le contenu du projet md2gemini dans un fichier appelé `md2gemini.zip`.

```
unpacked $ zip -r9 ../md2gemini *
```

Sortie

```
  adding: CJKwrap-2.2.dist-info/ (stored 0%)
  adding: CJKwrap-2.2.dist-info/RECORD (deflated 37%)
```

```
  ....
  adding: wcwidth-0.2.5.dist-info/INSTALLER (stored 0%)
  adding: wcwidth-0.2.5.dist-info/METADATA (deflated 62%)
```

6. (Facultatif) Utilisez les commandes suivantes pour vérifier que la bibliothèque fonctionne sur votre ordinateur local.

   a. Définissez le chemin Python vers l'emplacement du fichier `md2gemini.zip` et démarrez Python.

   ```
   /home $ PYTHONPATH=/tmp/testmd2gemini/md2gemini.zip
   /home python3
   ```

   b. Importez la bibliothèque et exécutez un test.

   ```
   >>> from md2gemini import md2gemini
   >>> print(md2gemini('[abc](https://abc.def)'))
   ```

   Sortie

   ```
   https://abc.def abc
   ```

7. Utilisez les commandes suivantes pour ajouter le fichier `.zip` à partir d'Amazon S3, l'importer dans votre bloc-notes dans Athena et effectuer un test non UDF.

```
# (non udf test)
sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/md2gemini.zip')
from md2gemini import md2gemini
print(md2gemini('[abc](https://abc.def)'))
```

Sortie

```
Calculation started (calculation_id=0ac0a082-6c3f-5a8f-eb6e-f8e9a5f9bc44) in
 (session=36c0a082-5338-3755-9f41-0cc954c55b35). Checking calculation status...
Calculation completed.
=> https://abc.def (https://abc.def/) abc
```

8. Utilisez les commandes suivantes pour effectuer un test UDF.

```
# (udf test)
```

```
from pyspark.sql.functions import udf
from pyspark.sql.functions import col
from md2gemini import md2gemini


hi_udf = udf(md2gemini)
df = spark.createDataFrame([(1, "[first website](https://abc.def)"), (2, "[second
 website](https://aws.com)")])
df.withColumn("col", hi_udf(col('_2'))).show()
```

Sortie

```
Calculation started (calculation_id=60c0a082-f04d-41c1-a10d-d5d365ef5157) in
 (session=36c0a082-5338-3755-9f41-0cc954c55b35). Checking calculation status...
Calculation completed.
+---+--------------------+--------------------+
| _1|                  _2|                 col|
+---+--------------------+--------------------+
|  1|[first website](h...|=> https://abc.de...|
|  2|[second website](...|=> https://aws.co...|
+---+--------------------+--------------------+
```

Ajout de fichiers JAR et de la configuration personnalisée de Spark

Lorsque vous créez ou modifiez une session dans Amazon Athena pour Apache Spark, vous pouvez utiliser les propriétés Spark pour spécifier des fichiers `.jar`, des packages ou une autre configuration personnalisée pour la session. Pour spécifier vos propriétés Spark, vous pouvez utiliser la console Athena, l'AWS CLI, ou l'API Athena.

Utilisation de la console Athena pour spécifier les propriétés Spark

Dans la console Athena, vous pouvez spécifier vos propriétés Spark lorsque vous créez un bloc-notes ou modifiez une session en cours.

Pour ajouter des propriétés dans la boîte de dialogue Créer un bloc-notes ou Modifier les détails de la session

1. Développez les propriétés Spark.
2. Pour ajouter vos propriétés, utilisez l'option Modifier dans la table ou Modifier dans JSON.

- Pour l'option Modifier dans la table, choisissez Ajouter une propriété pour ajouter une propriété, ou choisissez Supprimer pour supprimer une propriété. Utilisez les champs Clé et Valeur pour saisir les noms des propriétés et leurs valeurs.
  - Pour ajouter un fichier `.jar` personnalisé, utilisez la propriété `spark.jars`.
  - Utilisez la propriété `spark.jars.packages` pour spécifier un fichier de package.
- Pour saisir et modifier directement votre configuration, choisissez l'option Modifier dans JSON. Dans l'éditeur de texte JSON, vous pouvez effectuer les tâches suivantes :
  - Choisissez Copier pour copier le texte JSON dans le presse-papier.
  - Choisissez Effacer pour supprimer tout le texte de l'éditeur JSON.
  - Choisissez l'icône des paramètres (engrenage) pour configurer l'encapsulage des lignes ou choisissez un thème de couleur pour l'éditeur JSON.

Remarques

- Vous pouvez définir des propriétés dans Athena pour Spark, ce qui revient à définir les propriétés Spark directement sur un objet SparkConf.
- Commencez toutes les propriétés Spark par le préfixe `spark.`. Les propriétés comportant d'autres préfixes sont ignorées.
- Les propriétés Spark ne sont pas toutes disponibles pour une configuration personnalisée sur Athena. Si vous soumettez une demande `StartSession` dont la configuration est restreinte, la session ne démarre pas.
  - Vous ne pouvez pas utiliser le préfixe `spark.athena.` car il est réservé.

Utilisation de l'AWS CLI ou de l'API Athena pour fournir une configuration personnalisée

Pour utiliser l'AWS CLI ou l'API Athena afin de configurer votre session, utilisez l'action d'API StartSession ou la commande CLI start-session. Dans votre demande `StartSession`, utilisez le champ `SparkProperties` de l'objet EngineConfiguration pour transmettre vos informations de configuration au format JSON. Cela démarre une session avec la configuration spécifiée. Pour connaître la syntaxe des demandes, consultez StartSession dans la Référence d'API Amazon Athena.

Résolution des erreurs de démarrage de session

Lorsqu'une erreur de configuration personnalisée se produit lors du démarrage d'une session, la console Athena pour Spark affiche une bannière de message d'erreur. Pour résoudre les erreurs de démarrage de session, vous pouvez vérifier le changement d'état de la session ou les informations de journalisation.

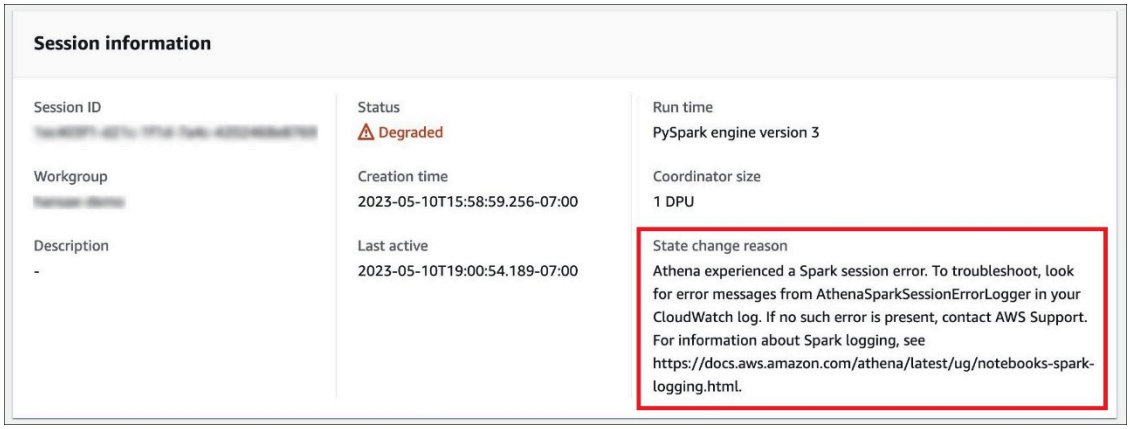
Affichage des informations de changement d'état de la session

Vous pouvez obtenir des informations sur un changement d'état de session à partir de l'éditeur de bloc-notes Athena ou de l'API Athena.

Pour afficher les informations sur l'état de session dans la console Athena

1. Dans l'éditeur de bloc-notes Athena, dans le menu Session en haut à droite, choisissez Afficher les détails.
2. Consultez l'onglet Session en cours. La section Informations de session affiche des informations telles que l'ID de session, le groupe de travail, le statut et le motif du changement d'état.

   L'exemple de capture d'écran suivant montre les informations contenues dans la section Motif du changement d'état de la boîte de dialogue Informations de session pour une erreur de session Spark dans Athena.



Pour afficher les informations sur l'état de la session à l'aide de l'API Athena

- Dans l'API Athena, vous pouvez trouver les informations de changement d'état de la session dans le champ `StateChangeReason` de l'objet SessionStatus.

> **Note**
>
> Après avoir arrêté manuellement une session, ou si la session s'arrête après un délai d'inactivité (la valeur par défaut est de 20 minutes), la valeur de StateChangeReason passe à La session a été arrêtée par demande.

Utilisation de la journalisation pour résoudre les erreurs de démarrage de session

Les erreurs de configuration personnalisées qui se produisent lors du démarrage d'une session sont journalisées par Amazon CloudWatch. Dans vos journaux CloudWatch Logs, recherchez les messages d'erreur provenant de `AthenaSparkSessionErrorLogger` pour résoudre les problèmes liés à l'échec du démarrage d'une session.

Pour plus d'informations sur la journalisation Spark, consultez Journalisation des événements de l'application Spark dans Athena.

Pour plus d'informations sur la résolution des problèmes des sessions dans Athena pour Spark, consultez Résolution des problèmes liés aux sessions.

Formats de données et de stockage pris en charge

Le tableau suivant présente les formats pris en charge en mode natif dans Athena pour Apache Spark.

Format de données	Lecture	Write (Écrire)	Compression d'écriture
parquet	oui	oui	aucun, non compressé, snappy, gzip
orc	oui	oui	aucun, snappy, zlib, lzo
json	oui	oui	bzip2, gzip, deflate
csv	oui	oui	bzip2, gzip, deflate

Format de données	Lecture	Write (Écrire)	Compression d'écriture
text	oui	oui	aucun, bzip2, gzip, deflate
fichier binaire	oui	N/A	N/A

Surveillance des calculs Apache Spark à l'aide des métriques CloudWatch

Athena publie les métriques liées aux calculs sur Amazon CloudWatch lorsque l'option Publish CloudWatch metrics de votre groupe de travail compatible avec Spark est sélectionnée. Dans la console CloudWatch, vous pouvez créer des tableaux de bord personnalisés et définir des alarmes et des déclencheurs pour les métriques.

Athena publie la métrique suivante dans la console CloudWatch sous l'espace de noms `AmazonAthenaForApacheSpark` :

- `DPUCount` – nombre de DPU consommés au cours de la session pour exécuter les calculs.

Cette métrique a les dimensions suivantes :

- `SessionId` – L'ID de la session dans laquelle les calculs sont soumis.
- `WorkGroup` – nom du groupe de travail.

Affichage des métriques des groupes de travail Spark dans la console Amazon CloudWatch

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation, sélectionnez Métriques, Toutes les métriques.
3. Sélectionnez l'espace de noms AmazonAthenaForApacheSpark.

Pour afficher les métriques grâce à la CLI

- Effectuez l'une des actions suivantes :

- Pour répertorier les métriques des groupes de travail compatibles avec Athena Spark, ouvrez une invite de commande et utilisez la commande suivante :

```
aws cloudwatch list-metrics --namespace "AmazonAthenaForApacheSpark"
```

- Pour répertorier toutes les métriques disponibles, utilisez la commande suivante :

```
aws cloudwatch list-metrics
```

Liste des métriques et dimensions CloudWatch pour les calculs Apache Spark dans Athena

Si vous avez activé les métriques CloudWatch dans votre groupe de travail Athena compatible avec Spark, Athena envoie les métriques suivantes à CloudWatch pour chaque groupe de travail. La métrique utilise l'espace de noms `AmazonAthenaForApacheSpark`.

Nom de métrique	Description
DPUCount	Nombre de DPU (unités de traitement de données) consommés pendant la session pour exécuter les calculs. Une DPU est une mesure relative de la puissance de traitement consistant en 4 vCPU de capacité de calcul et 16 Go de mémoire.

Cette métrique a les dimensions suivantes.

Dimension	Description
SessionId	L'ID de la session dans laquelle les calculs sont soumis.
WorkGroup	Le nom du groupe de travail.

Activation des compartiments Amazon S3 de type Paiement par le demandeur dans Athena pour Spark

Lorsqu'un compartiment Amazon S3 a une configuration de type Paiement par le demandeur, le compte de l'utilisateur exécutant la requête est débité pour les frais d'accès aux données et de transfert de données associés à la requête. Pour plus d'informations, consultez Utilisation des compartiments de type Paiement par le demandeur pour les transferts et l'utilisation du stockage dans le Guide de l'utilisateur Amazon S3.

Dans Athena pour Spark, les compartiments de type Paiement par le demandeur sont activés par session, et non par groupe de travail. De manière générale, l'activation des compartiments de type Paiement par le demandeur comprend les étapes suivantes :

1. Dans la console Amazon S3, activez le paiement par le demandeur sur les propriétés du compartiment et ajoutez une politique de compartiment pour spécifier l'accès.
2. Dans la console IAM, créez une politique IAM pour autoriser l'accès au compartiment, puis attachez la politique au rôle IAM qui sera utilisé pour accéder au compartiment de type Paiement par le demandeur.
3. Dans Athena pour Spark, ajoutez une propriété de session pour activer la fonction de paiement par le demandeur.

1. Activez le paiement par le demandeur sur un compartiment Amazon S3 et ajoutez une politique de compartiment

Pour activer le paiement par le demandeur sur un compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Dans la liste des compartiments, choisissez le lien du compartiment pour lequel vous souhaitez activer le paiement par le demandeur.
3. Sur la page du compartiment, choisissez l'onglet Propriétés.
4. Faites défiler jusqu'à la section Paiement par le demandeur, puis choisissez Modifier.
5. Sur la page Modifier le paiement par le demandeur, choisissez Activer, puis Enregistrer les modifications.
6. Choisissez l'onglet Permissions (Autorisations).
7. Dans la section Bucket policy (Politique de compartiment), sélectionnez Edit (Modifier).

8. Sur la page Modifier la politique de compartiment, appliquez la politique de compartiment que vous souhaitez au compartiment source. L'exemple de politique suivant donne accès à tous les principaux AWS (`"AWS": "*"`), mais votre accès peut être plus précis. Par exemple, vous pouvez souhaiter spécifier uniquement un rôle IAM spécifique dans un autre compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::account_number-us-east-1-my-s3-requester-pays-
bucket",
                "arn:aws:s3:::account_number-us-east-1-my-s3-requester-pays-bucket/
*"
            ]
        }
    ]
}
```

2. Créez une politique IAM et attachez-la à un rôle IAM

Ensuite, vous créez une politique IAM afin d'autoriser l'accès au compartiment. Vous attachez ensuite la politique au rôle qui sera utilisé pour accéder au compartiment de type Paiement par le demandeur.

Pour créer une politique IAM pour le compartiment de type Paiement par le demandeur et l'attacher à un rôle

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation de la console IAM, sélectionnez Politiques.
3. Sélectionnez Créer une politique.
4. Choisissez JSON.
5. Dans Éditeur de politiques, ajoutez une politique comme suit :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:*"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::account_number-us-east-1-my-s3-requester-pays-
bucket",
                "arn:aws:s3:::account_number-us-east-1-my-s3-requester-pays-bucket/
*"
            ]
        }
    ]
}
```

6. Choisissez Suivant.
7. Dans la page Vérifier et créer, entrez un nom et une Description facultative pour la politique, puis choisissez Créer une politique.
8. Dans le panneau de navigation, choisissez Roles (Rôles).
9. Sur la page Rôles, recherchez le rôle que vous souhaitez utiliser, puis cliquez sur le lien du nom du rôle.
10. Dans la section Politiques d'autorisations, choisissez Ajouter des autorisations, Attacher des politiques.
11. Dans la section Autres politiques d'autorisations, cochez la case correspondant à la politique que vous avez créée, puis choisissez Ajouter des autorisations.

3. Ajouter une propriété de session Athena pour Spark

Après avoir configuré le compartiment Amazon S3 et les autorisations associées pour le paiement par le demandeur, vous pouvez activer cette fonctionnalité dans une session Athena pour Spark.

Pour activer des compartiments de type Paiement par le demandeur dans une session Athena pour Spark

1. Dans l'éditeur de bloc-notes, dans le menu Session en haut à droite, choisissez Modifier la session.
2. Développez les propriétés Spark.
3. Choisissez Modifier dans JSON.
4. Dans l'éditeur de texte JSON, entrez ce qui suit :

```
{
  "spark.hadoop.fs.s3.useRequesterPaysHeader":"true"
}
```

5. Choisissez Enregistrer.

Activation du chiffrement Apache Spark

Vous pouvez activer le chiffrement Apache Spark dans Athena. Cela chiffre les données en transit entre les nœuds Spark et chiffre également les données au repos stockées localement par Spark. Pour renforcer la sécurité de ces données, Athena utilise la configuration de chiffrement suivante :

```
spark.io.encryption.keySizeBits="256"
spark.io.encryption.keygen.algorithm="HmacSHA384"
```

Pour activer le chiffrement Spark, vous pouvez utiliser la console Athena, l'AWS CLI ou l'API Athena.

Utilisation de la console Athena pour activer le chiffrement Spark

Pour créer un nouveau bloc-notes sur lequel le chiffrement Spark est activé

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.
3. Effectuez l'une des actions suivantes :

   - Dans Notebook explorer (Explorateur de blocs-notes), choisissez Create notebook (Créer un bloc-notes).

- Dans Notebook editor (Éditeur de bloc-notes), choisissez Create notebook (Créer un bloc-notes) ou cliquez sur l'icône plus (+) pour ajouter un bloc-notes.

4. Dans Nom du bloc-notes, entrez le nom du bloc-notes.
5. Développez l'option Propriétés Spark.
6. Sélectionnez Activer le chiffrement Spark.
7. Sélectionnez Créer.

La session de bloc-notes que vous créez est chiffrée. Utilisez le nouveau bloc-notes comme vous le feriez normalement. Lorsque vous lancerez ultérieurement de nouvelles sessions utilisant le bloc-notes, les nouvelles sessions seront également chiffrées.

Vous pouvez également utiliser la console Athena pour activer le chiffrement Spark sur un bloc-notes existant.

Pour activer le chiffrement sur un bloc-notes existant

1. Ouvrez une nouvelle session pour un bloc-notes créé précédemment.
2. Dans l'éditeur de bloc-notes, dans le menu Session en haut à droite, choisissez Modifier la session.
3. Dans la boîte de dialogue Modifier les détails de la session, développez Propriétés Spark.
4. Sélectionnez Activer le chiffrement Spark.
5. Choisissez Enregistrer.

La console lance une nouvelle session dont le chiffrement est activé. Le chiffrement sera également activé pour les sessions ultérieures que vous créerez pour ce bloc-notes.

Utilisation de l'AWS CLI pour activer le chiffrement Spark

Vous pouvez utiliser l'AWS CLI pour activer le chiffrement lorsque vous lancez une session en spécifiant les propriétés Spark appropriées.

Pour utiliser l'AWS CLI afin d'activer le chiffrement Spark

1. Utilisez une commande comme celle-ci pour créer un objet JSON de configuration du moteur qui spécifie les propriétés de chiffrement Spark.

```
ENGINE_CONFIGURATION_JSON=$(
```

```
  cat <<EOF
{
    "CoordinatorDpuSize": 1,
    "MaxConcurrentDpus": 20,
    "DefaultExecutorDpuSize": 1,
    "SparkProperties": {
      "spark.authenticate": "true",
      "spark.io.encryption.enabled": "true",
      "spark.network.crypto.enabled": "true"
    }
}
EOF
)
```

2. Dans l'AWS CLI, utilisez la commande `athena start-session` et transmettez l'objet JSON que vous avez créé à l'argument `--engine-configuration`, comme dans l'exemple suivant :

```
aws athena start-session \
   --region "region" \
   --work-group "your-work-group" \
   --engine-configuration "$ENGINE_CONFIGURATION_JSON"
```

Utilisation de l'API Athena pour activer le chiffrement Spark

Pour activer le chiffrement Spark avec l'API Athena, utilisez l'action StartSession et son paramètre `SparkProperties` EngineConfiguration pour spécifier la configuration de chiffrement dans votre demande `StartSession`.

Configuration de l' AWS Glue accès entre comptes dans Athena pour Spark

Cette rubrique explique comment le compte consommateur *666666666666* et le compte propriétaire *999999999999* peuvent être configurés pour un accès intercompte à AWS Glue . Lorsque les comptes sont configurés, le compte client peut exécuter des requêtes depuis Athena pour Spark sur les AWS Glue bases de données et les tables du propriétaire.

1. Dans AWS Glue, donnez accès aux rôles des consommateurs

Dans AWS Glue, le propriétaire crée une politique qui permet aux rôles du consommateur d'accéder au catalogue de AWS Glue données du propriétaire.

Pour ajouter une AWS Glue politique qui autorise un rôle de consommateur à accéder au catalogue de données du propriétaire

1. À l'aide du compte du propriétaire du catalogue, connectez-vous à la AWS Management Console.
2. Ouvrez la AWS Glue console à l'adresse https://console.aws.amazon.com/glue/.
3. Dans le panneau de navigation, développez Catalogue de données, puis choisissez Paramètres du catalogue.
4. Sur la page des paramètres du catalogue de données, dans la section Autorisations, ajoutez une politique similaire à la suivante. Cette politique prévoit des rôles pour le compte consommateur *666666666666* pour accéder au catalogue de données du compte propriétaire *999999999999*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Cataloguers",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::666666666666:role/Admin",
          "arn:aws:iam::666666666666:role/AWSAthenaSparkExecutionRole"
        ]
      },
      "Action": "glue:*",
      "Resource": [
        "arn:aws:glue:us-west-2:999999999999:catalog",
        "arn:aws:glue:us-west-2:999999999999:database/*",
        "arn:aws:glue:us-west-2:999999999999:table/*"
      ]
    }
  ]
}
```

2. Configurez le compte consommateur pour l'accès

Dans le compte client, créez une politique pour autoriser l'accès au propriétaire AWS Glue Data Catalog, aux bases de données et aux tables, et associez la politique à un rôle. L'exemple suivant utilise le compte consommateur *666666666666*.

Pour créer une AWS Glue politique d'accès au AWS Glue Data Catalog

1. À l'aide du compte consommateur, connectez-vous à la AWS Management Console.
2. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
3. Dans le panneau de navigation, développez Gestion des accès, puis choisissez Politiques.
4. Choisissez Créer une politique.
5. Sur la page Spécifier les autorisations, choisissez JSON.
6. Dans l'éditeur de politiques, entrez une instruction JSON comme la suivante qui autorise AWS Glue des actions sur le catalogue de données du compte propriétaire.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "glue:*",
            "Resource": [
                "arn:aws:glue:us-east-1:999999999999:catalog",
                "arn:aws:glue:us-east-1:999999999999:database/*",
                "arn:aws:glue:us-east-1:999999999999:table/*"
            ]
        }
    ]
}
```

7. Choisissez Suivant.
8. Sur la page Examiner et créer, dans Nom de la politique, saisissez un nom pour la politique.
9. Choisissez Créer une politique.

Ensuite, vous utilisez la console IAM du compte consommateur pour attacher la politique que vous venez de créer au ou aux rôles IAM que le compte consommateur utilisera pour accéder au catalogue de données du propriétaire.

Pour associer la AWS Glue politique aux rôles du compte client

1. Dans le panneau de navigation de la console IAM du compte consommateur, sélectionnez Rôles.
2. Sur la page Rôles, recherchez le rôle auquel vous souhaitez attacher la politique.
3. Choisissez Ajouter des autorisations, puis Attacher des politiques.
4. Recherchez la politique que vous venez de créer.
5. Cochez la case correspondant à la politique, puis choisissez Ajouter des autorisations.
6. Répétez les étapes pour ajouter la politique aux autres rôles que vous souhaitez utiliser.

3. Configurez une session et créez une requête

Dans Athena Spark, dans le compte du demandeur, à l'aide du rôle spécifié, créez une session pour tester l'accès en créant un bloc-notes ou en modifiant une session en cours. Lorsque vous configurez les propriétés de session, spécifiez l'une des options suivantes :

- Le séparateur de catalogue Glue : avec cette approche, vous incluez l'ID de compte propriétaire dans vos requêtes. Utilisez cette méthode si vous comptez utiliser la session pour interroger des catalogues de données provenant de différents propriétaires.
- L'ID du catalogue Glue : avec cette approche, vous interrogez directement la base de données. Cette méthode est plus pratique si vous comptez utiliser la session pour interroger uniquement le catalogue de données d'un seul propriétaire.

Utilisation de l'approche du séparateur de AWS Glue catalogue

Lorsque vous modifiez les propriétés de session, ajoutez les éléments suivants :

```
{
    "spark.hadoop.aws.glue.catalog.separator": "/"
}
```

Lorsque vous exécutez une requête dans une cellule, utilisez une syntaxe similaire à celle décrite dans l'exemple suivant. Notez que dans la clause `FROM`, l'ID du catalogue et le séparateur sont requis devant le nom de la base de données.

```
df = spark.sql('SELECT requestip, uri, method, status FROM `999999999999/
mydatabase`.cloudfront_logs LIMIT 5')
```

```
df.show()
```

Utilisation de l'approche des identifiants de AWS Glue catalogue

Lorsque vous modifiez les propriétés de session, entrez la propriété suivante. Remplacez *999999999999* par l'ID du compte propriétaire.

```
{
    "spark.hadoop.hive.metastore.glue.catalogid": "999999999999"
}
```

Lorsque vous exécutez une requête dans une cellule, utilisez une syntaxe similaire à celle ci-dessous. Notez que dans la clause `FROM`, l'ID du catalogue et le séparateur ne sont pas requis devant le nom de la base de données.

```
df = spark.sql('SELECT * FROM mydatabase.cloudfront_logs LIMIT 10')
df.show()
```

Ressources supplémentaires

Accès entre comptes aux catalogues de données AWS Glue

Gérer les autorisations entre comptes à l'aide des deux AWS Glue et de Lake Formation dans le guide du AWS Lake Formation développeur.

Configurez l'accès entre comptes à un partage à AWS Glue Data Catalog l'aide d'Amazon Athena AWS dans Prescriptive Guidance Patterns.

Service Quotas d'Amazon Athena pour Apache Spark

Les Service Quotas, également appelés limites, représentent le nombre maximal de ressources ou d'opérations de service que votre Compte AWS peut utiliser. Pour en savoir plus sur les Service Quotas d'autres services AWS susceptibles d'être utilisés avec Amazon Athena pour Spark, consultez AWS Service Quotas dans le Référence générale d'Amazon Web Services.

> **Note**
>
> Les nouveaux Comptes AWS peuvent avoir des quotas initiaux inférieurs, susceptibles d'augmenter au fil du temps. Amazon Athena pour Apache Spark surveille en permanence

l'utilisation du compte dans chaque Région AWS, puis augmente automatiquement les quotas en fonction de votre utilisation. Si vos exigences dépassent les limites indiquées, contactez le service client.

Le tableau suivant répertorie les Service Quotas pour Amazon Athena pour Apache Spark.

Nom	Par défaut	Ajustable	Description
Simultanéité DPU Apache Spark	160	Non	Nombre maximum d'unités de traitement de données (DPU) que vous pouvez utiliser simultanément pour les calculs d'Apache Spark pour un seul compte dans la Région AWS actuelle. Une DPU est une mesure relative de la puissance de traitement consistant en 4 vCPU de capacité de calcul et 16 Go de mémoire.
Simultanéité DPU de session Apache Spark	60	Non	Nombre maximum de DPU que vous pouvez utiliser simultanément pour un calcul Apache Spark au cours d'une session.

API de bloc-notes Athena

La liste suivante contient des liens de référence vers les actions des API de bloc-notes Athena. Pour les structures de données et les autres actions des API Athena, voir la Référence d'API Amazon Athena.

- CreateNotebook
- CreatePresignedNotebookUrl
- DeleteNotebook
- ExportNotebook
- GetCalculationExecution
- GetCalculationExecutionCode

- GetCalculationExecutionStatus
- GetNotebookMetadata
- GetSession
- GetSessionStatus
- ImportNotebook
- ListApplicationDPUSizes
- ListCalculationExecutions
- ListExecutors
- ListNotebookMetadata
- ListNotebookSessions
- ListSessions
- StartCalculationExecution
- StartSession
- StopCalculationExecution
- TerminateSession
- UpdateNotebook
- UpdateNotebookMetadata

Problèmes connus dans Athena pour Spark

Cette page présente certains des problèmes connus d'Athena pour Apache Spark.

Exception d'argument non valide lors de la création d'une table

Bien que Spark n'autorise pas la création de bases de données avec une propriété d'emplacement vide, les bases de données AWS Glue peuvent avoir une `LOCATION` propriété vide si elles sont créées en dehors de Spark.

Si vous créez une table et spécifiez une AWS Glue base de données contenant un `LOCATION` champ vide, une exception comme celle-ci peut se produire IllegalArgumentException: Impossible de créer un chemin à partir d'une chaîne vide.

Par exemple, la commande suivante génère une exception si la base de données par défaut de AWS Glue contient un champ `LOCATION` vide :

```
spark.sql("create table testTable (firstName STRING)")
```

Solution suggérée A — AWS Glue À utiliser pour ajouter un emplacement à la base de données que vous utilisez.

Pour ajouter un emplacement à une AWS Glue base de données

1. Connectez-vous à la AWS Glue console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/glue/.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Dans la liste des bases de données, choisissez la base de données à modifier.
4. Sur la page de détails de la base de données, sélectionnez Edit (Modifier).
5. Sur la page Update a database (Mettre à jour une base de données), dans le champ Location (Emplacement), saisissez un emplacement Amazon S3.
6. Choisissez Update Database (Mettre à jour la base de données).

Solution suggérée B – Utilisez une autre base de données AWS Glue qui a un emplacement existant et valide dans Amazon S3. Par exemple, si vous avez une base de données appelée `dbWithLocation`, utilisez la commande `spark.sql("use dbWithLocation")` pour passer à cette base de données.

Solution suggérée C – Lorsque vous utilisez Spark SQL pour créer la table, spécifiez une valeur pour `location`, comme dans l'exemple suivant.

```
spark.sql("create table testTable (firstName STRING)
       location 's3://DOC-EXAMPLE-BUCKET/'").
```

Solution suggérée D – Si vous avez indiqué un emplacement lors de la création de la table, mais que le problème persiste, assurez-vous que le chemin Amazon S3 que vous indiquez comporte une barre oblique à la fin. Par exemple, la commande suivante génère une exception d'argument non valide :

```
spark.sql("create table testTable (firstName STRING)
       location 's3://DOC-EXAMPLE-BUCKET'")
```

Pour corriger cela, ajoutez une barre oblique finale à l'emplacement (par exemple, `'s3:// DOC-EXAMPLE-BUCKET/'`).

Base de données créée dans un emplacement de groupe de travail

Si vous utilisez une commande comme `spark.sql('create database db')` pour créer une base de données et que vous ne spécifiez pas d'emplacement pour la base de données, Athena crée un sous-répertoire dans l'emplacement de votre groupe de travail et utilise cet emplacement pour la base de données nouvellement créée.

Problèmes liés aux tables gérées par Hive dans la base de données AWS Glue par défaut

Si la `Location` propriété de votre base de données par défaut AWS Glue n'est pas vide et indique un emplacement valide dans Amazon S3, et que vous utilisez Athena for Spark pour créer une table gérée par Hive dans AWS Glue votre base de données par défaut, les données sont écrites à l'emplacement Amazon S3 spécifié dans votre groupe de travail Athena Spark plutôt qu'à l'emplacement spécifié par la base de données. AWS Glue

Ce problème se produit en raison de la façon dont Apache Hive gère sa base de données par défaut. Apache Hive crée des données de table dans l'emplacement racine de l'entrepôt Hive, qui peut être différent de l'emplacement de base de données par défaut réel.

Lorsque vous utilisez Athena pour Spark pour créer une table gérée par Hive sous la base de données par défaut dans AWS Glue, les métadonnées de la AWS Glue table peuvent pointer vers deux emplacements différents. Cela peut provoquer un comportement inattendu lorsque vous tentez une opération `INSERT` ou `DROP TABLE`.

Les étapes pour reproduire le problème sont les suivantes :

1. Dans Athena pour Spark, vous utilisez l'une des méthodes suivantes pour créer ou enregistrer une table gérée par Hive :
   - Une instruction SQL telle que `CREATE TABLE $tableName`
   - Une telle PySpark commande `df.write.mode("overwrite").saveAsTable($tableName)` ne spécifie pas l'`path`option dans l'API Dataframe.

   À ce stade, la AWS Glue console peut indiquer un emplacement incorrect dans Amazon S3 pour la table.
2. Dans Athena pour Spark, vous utilisez l'instruction `DROP TABLE $table_name` pour supprimer la table que vous avez créée.

3. Après avoir exécuté l'instruction `DROP TABLE`, vous remarquez que les fichiers sous-jacents sont toujours présents dans Amazon S3.

Pour résoudre ce problème, procédez de l'une des manières suivantes :

Solution A — Utilisez une autre AWS Glue base de données lorsque vous créez des tables gérées par Hive.

Solution B : spécifiez un emplacement vide pour la base de données par défaut dans AWS Glue. Créez ensuite vos tables gérées dans la base de données par défaut.

Incompatibilité des formats de fichier CSV et JSON entre Athena pour Spark et Athena SQL

En raison d'un problème connu lié à Spark open source, lorsque vous créez une table dans Athena pour Spark sur des données CSV ou JSON, la table peut ne pas être lisible à partir d'Athena SQL, et vice versa.

Par exemple, vous pouvez créer une table dans Athena pour Spark de l'une des manières suivantes :

- Utilisez la syntaxe `USING csv` suivante :

```
spark.sql('''CREATE EXTERNAL TABLE $tableName (
$colName1 $colType1,
$colName2 $colType2,
$colName3 $colType3)
USING csv
PARTITIONED BY ($colName1)
LOCATION $s3_location''')
```

- Avec la syntaxe DataFrameAPI suivante :

```
df.write.format('csv').saveAsTable($table_name)
```

En raison du problème connu avec Spark open source, les requêtes d'Athena SQL sur les tables résultantes risquent d'échouer.

Solution suggérée : essayez de créer la table dans Athena pour Spark en utilisant la syntaxe Apache Hive. Pour plus d'informations, consultez CREATE HIVEFORMAT TABLE dans la documentation Apache Spark.

Résolution des problèmes liés à Athena pour Spark

Utilisez les informations suivantes pour résoudre les problèmes que vous pouvez rencontrer lorsque vous utilisez des blocs-notes et des sessions sur Athena.

Rubriques

Résolution des problèmes liés aux groupes de travail compatibles avec Spark

Consultez les informations suivantes pour tenter de résoudre les problèmes liés aux groupes de travail compatibles avec Spark dans Athena.

La session cesse de répondre lors de l'utilisation d'un rôle IAM existant

Si vous n'avez pas créé un nouveau `AWSAthenaSparkExecutionRole` pour votre groupe de travail compatible avec Spark et que vous avez plutôt mis à jour ou choisi un rôle IAM existant, il est possible que votre session cesse de répondre. Dans ce cas, vous devrez peut-être ajouter les politiques de confiance et d'autorisations suivantes à votre rôle d'exécution de groupe de travail compatible avec Spark.

Ajoutez l'exemple de politique de confiance suivant. La politique comprend un contrôle du député confus pour le rôle d'exécution. Remplacez les valeurs pour *`111122223333aws-region`*, et *`workgroup-name`* par l' Compte AWS ID et Région AWS le groupe de travail que vous utilisez.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "athena.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:athena:aws-
region:111122223333:workgroup/workgroup-name"
                }
            }
        }
    ]
}
```

Ajoutez une politique d'autorisations comme la politique par défaut suivante pour les groupes de travail compatibles avec les blocs-notes. Modifiez les emplacements et les Compte AWS identifiants Amazon S3 de l'espace réservé pour qu'ils correspondent à ceux que vous utilisez. Remplacez les valeurs de DOC-EXAMPLE-BUCKET, *aws-region*, *111122223333* et *workgroup-name* par l'ID du compartiment Amazon S3, de la Région AWS, du Compte AWS et le groupe de travail que vous utilisez.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
```

```
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:GetWorkGroup",
                "athena:CreatePresignedNotebookUrl",
                "athena:TerminateSession",
                "athena:GetSession",
                "athena:GetSessionStatus",
                "athena:ListSessions",
                "athena:StartCalculationExecution",
                "athena:GetCalculationExecutionCode",
                "athena:StopCalculationExecution",
                "athena:ListCalculationExecutions",
                "athena:GetCalculationExecution",
                "athena:GetCalculationExecutionStatus",
                "athena:ListExecutors",
                "athena:ExportNotebook",
                "athena:UpdateNotebook"
            ],
            "Resource": "arn:aws:athena:aws-region:111122223333:workgroup/workgroup-
name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:aws-region:111122223333:log-group:/aws-athena:*",
                "arn:aws:logs:aws-region:111122223333:log-group:/aws-athena*:log-
stream:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "logs:DescribeLogGroups",
            "Resource": "arn:aws:logs:aws-region:111122223333:log-group:*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "AmazonAthenaForApacheSpark"
                }
            }
        }
    ]
}
```

Utilisation de l'instruction Spark EXPLAIN pour résoudre les problèmes liés à Spark SQL

Vous pouvez utiliser l'instruction Spark `EXPLAIN` avec Spark SQL pour résoudre les problèmes liés à votre code Spark. Les exemples de code et de résultats suivants illustrent cette utilisation.

Example – Instruction Spark SELECT

```
spark.sql("select * from select_taxi_table").explain(True)
```

Sortie

```
Calculation started (calculation_id=20c1ebd0-1ccf-ef14-db35-7c1844876a7e) in
(session=24c1ebcb-57a8-861e-1023-736f5ae55386).
Checking calculation status...

Calculation completed.
== Parsed Logical Plan ==
'Project [*]
+- 'UnresolvedRelation [select_taxi_table], [], false

== Analyzed Logical Plan ==
VendorID: bigint, passenger_count: bigint, count: bigint
Project [VendorID#202L, passenger_count#203L, count#204L]
+- SubqueryAlias spark_catalog.spark_demo_database.select_taxi_table
   +- Relation spark_demo_database.select_taxi_table[VendorID#202L,
```

```
       passenger_count#203L,count#204L] csv

== Optimized Logical Plan ==
Relation spark_demo_database.select_taxi_table[VendorID#202L,
passenger_count#203L,count#204L] csv

== Physical Plan ==
FileScan csv spark_demo_database.select_taxi_table[VendorID#202L,
passenger_count#203L,count#204L]
Batched: false, DataFilters: [], Format: CSV,
Location: InMemoryFileIndex(1 paths)
[s3://DOC-EXAMPLE-BUCKET/select_taxi],
PartitionFilters: [], PushedFilters: [],
ReadSchema: struct<VendorID:bigint,passenger_count:bigint,count:bigint>
```

Example – Bloc de données Spark

L'exemple suivant illustre comment utiliser `EXPLAIN` avec un bloc de données Spark.

```
taxi1_df=taxi_df.groupBy("VendorID", "passenger_count").count()
taxi1_df.explain("extended")
```

Sortie

```
Calculation started (calculation_id=d2c1ebd1-f9f0-db25-8477-3effc001b309) in
(session=24c1ebcb-57a8-861e-1023-736f5ae55386).
Checking calculation status...

Calculation completed.
== Parsed Logical Plan ==
'Aggregate ['VendorID, 'passenger_count],
['VendorID, 'passenger_count, count(1) AS count#321L]
+- Relation [VendorID#49L,tpep_pickup_datetime#50,tpep_dropoff_datetime#51,
passenger_count#52L,trip_distance#53,RatecodeID#54L,store_and_fwd_flag#55,
PULocationID#56L,DOLocationID#57L,payment_type#58L,fare_amount#59,
extra#60,mta_tax#61,tip_amount#62,tolls_amount#63,improvement_surcharge#64,
total_amount#65,congestion_surcharge#66,airport_fee#67] parquet

== Analyzed Logical Plan ==
VendorID: bigint, passenger_count: bigint, count: bigint
Aggregate [VendorID#49L, passenger_count#52L],
[VendorID#49L, passenger_count#52L, count(1) AS count#321L]
+- Relation [VendorID#49L,tpep_pickup_datetime#50,tpep_dropoff_datetime#51,
```

```
passenger_count#52L,trip_distance#53,RatecodeID#54L,store_and_fwd_flag#55,
PULocationID#56L,DOLocationID#57L,payment_type#58L,fare_amount#59,extra#60,
mta_tax#61,tip_amount#62,tolls_amount#63,improvement_surcharge#64,
total_amount#65,congestion_surcharge#66,airport_fee#67] parquet

== Optimized Logical Plan ==
Aggregate [VendorID#49L, passenger_count#52L],
[VendorID#49L, passenger_count#52L, count(1) AS count#321L]
+- Project [VendorID#49L, passenger_count#52L]
   +- Relation [VendorID#49L,tpep_pickup_datetime#50,tpep_dropoff_datetime#51,
passenger_count#52L,trip_distance#53,RatecodeID#54L,store_and_fwd_flag#55,
PULocationID#56L,DOLocationID#57L,payment_type#58L,fare_amount#59,extra#60,
mta_tax#61,tip_amount#62,tolls_amount#63,improvement_surcharge#64,
total_amount#65,congestion_surcharge#66,airport_fee#67] parquet

== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- HashAggregate(keys=[VendorID#49L, passenger_count#52L], functions=[count(1)],
output=[VendorID#49L, passenger_count#52L, count#321L])
   +- Exchange hashpartitioning(VendorID#49L, passenger_count#52L, 1000),
      ENSURE_REQUIREMENTS, [id=#531]
      +- HashAggregate(keys=[VendorID#49L, passenger_count#52L],
         functions=[partial_count(1)], output=[VendorID#49L,
         passenger_count#52L, count#326L])
         +- FileScan parquet [VendorID#49L,passenger_count#52L] Batched: true,
            DataFilters: [], Format: Parquet,
            Location: InMemoryFileIndex(1 paths)[s3://DOC-EXAMPLE-BUCKET/
            notebooks/yellow_tripdata_2016-01.parquet], PartitionFilters: [],
            PushedFilters: [],
            ReadSchema: struct<VendorID:bigint,passenger_count:bigint>
```

Journalisation des événements de l'application Spark dans Athena

L'éditeur de bloc-notes Athena permet la journalisation standard sur Jupyter, Spark et Python. Vous pouvez l'utiliser `df.show()` pour afficher PySpark DataFrame le contenu ou `print("Output")` pour afficher des valeurs dans la sortie de la cellule. Les sorties `stdout`, `stderr` et `results` de vos calculs sont écrites dans votre compartiment de résultats de requête dans Amazon S3.

Enregistrement des événements de l'application Spark sur Amazon CloudWatch

Vos sessions Athena peuvent également écrire des journaux sur Amazon sur CloudWatch le compte que vous utilisez.

Compréhension des flux de journaux et des groupes de journaux

CloudWatch organise l'activité des journaux en flux de journaux et en groupes de journaux.

Flux de journaux : un flux de CloudWatch journal est une séquence d'événements de journal qui partagent la même source. Chaque source distincte de CloudWatch journaux dans Logs constitue un flux de journaux distinct.

Groupes de CloudWatch journaux : dans Logs, un groupe de journaux est un groupe de flux de journaux qui partagent les mêmes paramètres de conservation, de surveillance et de contrôle d'accès.

Le nombre de flux de journaux pouvant appartenir à un groupe de journaux est illimité.

Dans Athena, lorsque vous démarrez une session de bloc-notes pour la première fois, Athena crée un groupe de journaux CloudWatch qui utilise le nom de votre groupe de travail compatible avec Spark, comme dans l'exemple suivant.

```
/aws-athena/workgroup-name
```

Ce groupe de journaux reçoit un flux de journaux pour chaque exécuteur de votre session qui produit au moins un événement du journal. L'exécuteur est la plus petite unité de calcul qu'une session de bloc-notes peut demander à Athena. Dans CloudWatch, le nom du flux de journal commence par l'ID de session et l'ID de l'exécuteur.

Pour plus d'informations sur les groupes de CloudWatch journaux et les flux de journaux, consultez la section Utilisation des groupes de journaux et des flux de CloudWatch journaux dans le guide de l'utilisateur Amazon Logs.

Utilisation d'objets de journalisation standard dans Athena pour Spark

Dans une session Athena for Spark, vous pouvez utiliser les deux objets de journalisation standard mondiaux suivants pour écrire des journaux sur Amazon : CloudWatch

- athena_user_logger — Envoie les journaux uniquement à. CloudWatch Utilisez cet objet lorsque vous souhaitez enregistrer des informations directement dans lesquelles vos applications Spark sont CloudWatch enregistrées, comme dans l'exemple suivant.

  ```
  athena_user_logger.info("CloudWatch log line.")
  ```

L'exemple écrit un événement de journal CloudWatch comme suit :

```
AthenaForApacheSpark: 2022-01-01 12:00:00,000 INFO builtins: CloudWatch log line.
```

- athena_shared_logger — Envoie le même journal à et à des fins d'assistance. CloudWatch AWS Vous pouvez utiliser cet objet pour partager des journaux avec les équipes de AWS service à des fins de résolution des problèmes, comme dans l'exemple suivant.

```
athena_shared_logger.info("Customer debug line.")
var = [...some variable holding customer data...]
athena_shared_logger.info(var)
```

L'exemple enregistre la `debug` ligne et la valeur de la `var` variable dans CloudWatch Logs et envoie une copie de chaque ligne à AWS Support.

> **Note**
>
> Pour des raisons de confidentialité, votre code de calcul et vos résultats ne sont pas partagés avec AWS. Veillez à ce que vos appels à `athena_shared_logger` n'écrivent que les informations que vous voulez rendre visibles à AWS Support.

Les enregistreurs fournis écrivent des événements via Apache Log4j et héritent les niveaux de journalisation de cette interface. Les valeurs de niveau de journalisation possibles sont `DEBUG`, `ERROR`, `FATAL`, `INFO`, `WARN` et `WARNING`. Vous pouvez utiliser la fonction nommée correspondante sur l'enregistreur pour produire ces valeurs.

> **Note**
>
> Ne reliez pas les noms `athena_user_logger` ou `athena_shared_logger`. Cela empêche les objets de journalisation d'écrire CloudWatch pendant le reste de la session.

Exemple : enregistrement des événements du bloc-notes dans CloudWatch

La procédure suivante explique comment enregistrer les événements du bloc-notes Athena dans Amazon CloudWatch Logs.

Pour enregistrer les événements du bloc-notes Athena dans Amazon Logs CloudWatch

1. Suivez Démarrage avec Apache Spark sur Amazon Athena pour créer un groupe de travail compatible avec Spark dans Athena avec un nom unique. Ce tutoriel utilise le nom de groupe de travail `athena-spark-example`.
2. Suivez les étapes de Création de votre propre bloc-notes pour créer un bloc-notes et lancer une nouvelle session.
3. Dans l'éditeur de bloc-notes Athena, dans une nouvelle cellule de bloc-notes, saisissez la commande suivante :

   ```
   athena_user_logger.info("Hello world.")
   ```

4. Exécuter la cellule.
5. Récupérez l'ID de la session actuelle en effectuant l'une des opérations suivantes :

   - Affichez la sortie de la cellule (par exemple, `... session=72c24e73-2c24-8b22-14bd-443bdcd72de4`).
   - Dans une nouvelle cellule, exécutez la commande magique `%session_id`.
6. Enregistrer l'ID de la session.
7. Avec le même Compte AWS que celui que vous utilisez pour exécuter la session du bloc-notes, ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
8. Dans le volet de navigation de la CloudWatch console, sélectionnez Log groups.
9. Dans la liste des groupes de journaux, choisissez le groupe de journaux qui porte le nom de votre groupe de travail Athena compatible avec Spark, comme dans l'exemple suivant.

   ```
   /aws-athena/athena-spark-example
   ```

   La section Log streams (Flux de journaux) contient une liste d'un ou plusieurs liens de flux de journaux pour le groupe de travail. Le nom de chaque flux de journaux contient l'ID de la session, l'ID de l'exécuteur et l'UUID unique, séparés par des barres obliques.

   Par exemple, si l'ID de la session est `5ac22d11-9fd8-ded7-6542-0412133d3177` et l'ID de l'exécuteur est `f8c22d11-9fd8-ab13-8aba-c4100bfba7e2`, le nom du flux de journaux ressemble à l'exemple suivant.

```
5ac22d11-9fd8-ded7-6542-0412133d3177/f8c22d11-9fd8-ab13-8aba-c4100bfba7e2/f012d7cb-cefd-40b1-90b9-67358f003d0b
```

10. Choisissez le lien du flux de journaux pour votre session.
11. Sur la page Log events (Événements du journal), consultez la colonne Message.

    L'événement du journal pour la cellule que vous avez exécutée ressemble à ce qui suit :

    ```
    AthenaForApacheSpark: 2022-01-01 12:00:00,000 INFO builtins: Hello world.
    ```

12. Retournez à l'éditeur de blocs-notes Athena.
13. Dans une nouvelle cellule, saisissez le code suivant. Le code enregistre une variable pour CloudWatch :

    ```
    x = 6
    athena_user_logger.warn(x)
    ```

14. Exécuter la cellule.
15. Retournez à la page des événements du journal de la CloudWatch console pour le même flux de journal.
16. Le flux de journaux contient maintenant une entrée d'événement du journal avec un message comme le suivant :

    ```
    AthenaForApacheSpark: 2022-01-01 12:00:00,000 WARN builtins: 6
    ```

Utilisation de CloudTrail pour résoudre les problèmes liés aux appels API du bloc-notes Athena

Pour résoudre les problèmes liés aux appels API du bloc-notes, vous pouvez consulter les journaux Athena CloudTrail afin d'identifier les anomalies ou de découvrir les actions initiées par les utilisateurs. Pour des informations détaillées sur l'utilisation de CloudTrail avec Athena, voir Journalisation des appels d'API Amazon Athena avec AWS CloudTrail.

Les exemples suivants illustrent les entrées de journal CloudTrail pour les API du bloc-notes Athena :

- StartSession
- TerminateSession

- ImportNotebook
- UpdateNotebook
- StartCalculationExecution

StartSession

L'exemple suivant illustre le journal CloudTrail pour l'événement StartSession d'un bloc-notes.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE_PRINCIPAL_ID:alias",
        "arn": "arn:aws:sts::123456789012:assumed-role/Admin/alias",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLE_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-10-14T16:41:51Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-10-14T17:05:36Z",
    "eventSource": "athena.amazonaws.com",
    "eventName": "StartSession",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.10",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36",
    "requestParameters": {
        "workGroup": "notebook-workgroup",
        "engineConfiguration": {
            "coordinatorDpuSize": 1,
```

```
            "maxConcurrentDpus": 20,
            "defaultExecutorDpuSize": 1,
            "additionalConfigs": {
                "NotebookId": "b8f5854b-1042-4b90-9d82-51d3c2fd5c04",
                "NotebookIframeParentUrl": "https://us-east-1.console.aws.amazon.com"
            }
        },
        "notebookVersion": "KeplerJupyter-1.x",
        "sessionIdleTimeoutInMinutes": 20,
        "clientRequestToken": "d646ff46-32d2-42f0-94d1-d060ec3e5d78"
    },
    "responseElements": {
        "sessionId": "a2c1ebba-ad01-865f-ed2d-a142b7451f7e",
        "state": "CREATED"
    },
    "requestID": "d646ff46-32d2-42f0-94d1-d060ec3e5d78",
    "eventID": "b58ce998-eb89-43e9-8d67-d3d8e30561c9",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "athena.us-east-1.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```


TerminateSession

L'exemple suivant illustre le journal CloudTrail pour l'événement TerminateSession d'un bloc-notes.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE_PRINCIPAL_ID:alias",
        "arn": "arn:aws:sts::123456789012:assumed-role/Admin/alias",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "sessionContext": {
```

```
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLE_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-10-14T16:41:51Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-10-14T17:21:03Z",
    "eventSource": "athena.amazonaws.com",
    "eventName": "TerminateSession",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.11",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36",
    "requestParameters": {
        "sessionId": "a2c1ebba-ad01-865f-ed2d-a142b7451f7e"
    },
    "responseElements": {
        "state": "TERMINATING"
    },
    "requestID": "438ea37e-b704-4cb3-9a76-391997cf42ee",
    "eventID": "49026c5a-bf58-4cdb-86ca-978e711ad238",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "athena.us-east-1.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```

ImportNotebook

L'exemple suivant illustre le journal CloudTrail pour l'événement ImportNotebook d'un bloc-notes. Pour des raisons de sécurité, certains contenus sont cachés.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE_PRINCIPAL_ID:alias",
        "arn": "arn:aws:sts::123456789012:assumed-role/Admin/alias",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLE_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-10-14T16:41:51Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-10-14T17:08:54Z",
    "eventSource": "athena.amazonaws.com",
    "eventName": "ImportNotebook",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36",
    "requestParameters": {
        "workGroup": "notebook-workgroup",
        "name": "example-notebook-name",
        "payload": "HIDDEN_FOR_SECURITY_REASONS",
        "type": "IPYNB",
        "contentMD5": "HIDDEN_FOR_SECURITY_REASONS"
    },
    "responseElements": {
```

```
        "notebookId": "05f6225d-bdcc-4935-bc25-a8e19434652d"
    },
    "requestID": "813e777f-6dac-41f4-82a7-e99b7b33f319",
    "eventID": "4abec837-143b-4458-9c1f-fa9fb88ab69b",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "athena.us-east-1.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```


UpdateNotebook

L'exemple suivant illustre le journal CloudTrail pour l'événement UpdateNotebook d'un bloc-notes. Pour des raisons de sécurité, certains contenus sont cachés.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE_PRINCIPAL_ID:AthenaExecutor-9cc1ebb2-aac5-
b1ca-8247-5d827bd8232f",
        "arn": "arn:aws:sts::123456789012:assumed-role/AWSAthenaSparkExecutionRole-
om0yj71w5l/AthenaExecutor-9cc1ebb2-aac5-b1ca-8247-5d827bd8232f",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLE_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:role/service-role/
AWSAthenaSparkExecutionRole-om0yj71w5l",
                "accountId": "123456789012",
                "userName": "AWSAthenaSparkExecutionRole-om0yj71w5l"
            },
            "webIdFederationData": {},
            "attributes": {
```

```
                "creationDate": "2022-10-14T16:48:06Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-10-14T16:52:22Z",
    "eventSource": "athena.amazonaws.com",
    "eventName": "UpdateNotebook",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.13",
    "userAgent": "Boto3/1.24.84 Python/3.8.14 Linux/4.14.225-175.364.amzn2.aarch64
 Botocore/1.27.84",
    "requestParameters": {
        "notebookId": "c87553ff-e740-44b5-884f-a70e575e08b9",
        "payload": "HIDDEN_FOR_SECURITY_REASONS",
        "type": "IPYNB",
        "contentMD5": "HIDDEN_FOR_SECURITY_REASONS",
        "sessionId": "9cc1ebb2-aac5-b1ca-8247-5d827bd8232f"
    },
    "responseElements": null,
    "requestID": "baaba1d2-f73d-4df1-a82b-71501e7374f1",
    "eventID": "745cdd6f-645d-4250-8831-d0ffd2fe3847",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "athena.us-east-1.amazonaws.com"
    }
}
```

StartCalculationExecution

L'exemple suivant illustre le journal CloudTrail pour l'événement StartCalculationExecution d'un bloc-notes. Pour des raisons de sécurité, certains contenus sont cachés.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
```

```
        "principalId": "EXAMPLE_PRINCIPAL_ID:AthenaExecutor-9cc1ebb2-aac5-
b1ca-8247-5d827bd8232f",
        "arn": "arn:aws:sts::123456789012:assumed-role/AWSAthenaSparkExecutionRole-
om0yj71w5l/AthenaExecutor-9cc1ebb2-aac5-b1ca-8247-5d827bd8232f",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLE_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:role/service-role/
AWSAthenaSparkExecutionRole-om0yj71w5l",
                "accountId": "123456789012",
                "userName": "AWSAthenaSparkExecutionRole-om0yj71w5l"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-10-14T16:48:06Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-10-14T16:52:37Z",
    "eventSource": "athena.amazonaws.com",
    "eventName": "StartCalculationExecution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.14",
    "userAgent": "Boto3/1.24.84 Python/3.8.14 Linux/4.14.225-175.364.amzn2.aarch64
 Botocore/1.27.84",
    "requestParameters": {
        "sessionId": "9cc1ebb2-aac5-b1ca-8247-5d827bd8232f",
        "description": "Calculation started via Jupyter notebook",
        "codeBlock": "HIDDEN_FOR_SECURITY_REASONS",
        "clientRequestToken": "0111cd63-4fd0-4ad8-a738-fd350115fc21"
    },
    "responseElements": {
        "calculationExecutionId": "82c1ebb4-bd08-e4c3-5631-a662fb2ff2c5",
        "state": "CREATING"
    },
    "requestID": "1a107461-3f1b-481e-b8a2-7fbd524e2373",
    "eventID": "b74dbd00-e839-4bd1-a1da-b75fbc70ab9a",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

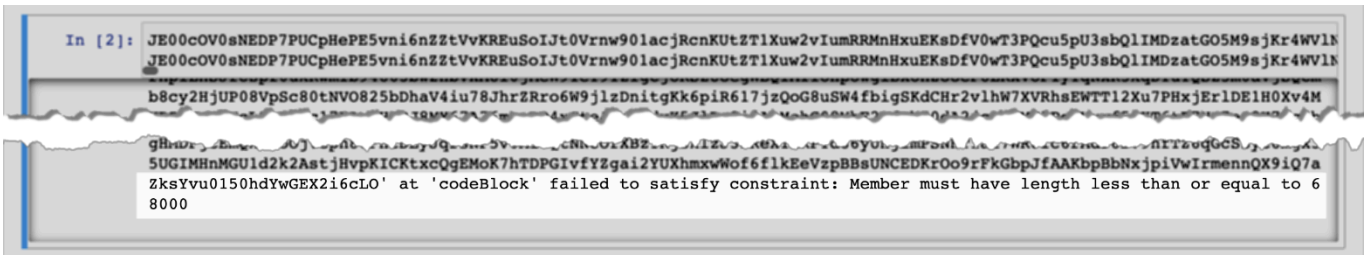
```
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "athena.us-east-1.amazonaws.com"
    }
}
```

Dépassement de la limite de taille des blocs de code de 68 000

Athena pour Spark a une limite de taille des blocs de code de calcul connue de 68 000 caractères. Lorsque vous exécutez un calcul avec un bloc de code dépassant cette limite, le message d'erreur suivant peut s'afficher :

«... » à « CodeBlock » ne satisfait pas à la contrainte : le membre doit avoir une longueur inférieure ou égale à 68000

L'image suivante montre cette erreur dans l'éditeur de bloc-notes de la console Athena.



La même erreur peut se produire lorsque vous utilisez l'AWS CLI pour exécuter un calcul comportant un bloc de code volumineux, comme dans l'exemple suivant.

```
aws athena start-calculation-execution \
    --session-id "{SESSION_ID}" \
    --description "{SESSION_DESCRIPTION}" \
    --code-block "{LARGE_CODE_BLOCK}"
```

La commande affiche le message d'erreur suivant :

{LARGE_CODE_BLOCK} à « CodeBlock » ne satisfait pas à la contrainte : le membre doit avoir une longueur inférieure ou égale à 68000

Solution

Pour contourner ce problème, chargez le fichier contenant votre code de requête ou de calcul sur Amazon S3. Utilisez ensuite boto3 pour lire le fichier et exécuter votre SQL ou votre code.

Les exemples suivants supposent que vous avez déjà chargé le fichier contenant votre requête SQL ou votre code Python sur Amazon S3.

Exemple SQL

L'exemple de code suivant lit le fichier `large_sql_query.sql` à partir d'un compartiment Amazon S3, puis exécute la requête volumineuse que le fichier contient.

```
s3 = boto3.resource('s3')
def read_s3_content(bucket_name, key):
    response = s3.Object(bucket_name, key).get()
    return response['Body'].read()

# SQL
sql = read_s3_content('bucket_name', 'large_sql_query.sql')
df = spark.sql(sql)
```

Exemple PySpark

L'exemple de code suivant lit le fichier `large_py_spark.py` à partir d'Amazon S3, puis exécute le bloc de code volumineux que le fichier contient.

```
s3 = boto3.resource('s3')

def read_s3_content(bucket_name, key):
    response = s3.Object(bucket_name, key).get()
    return response['Body'].read()

# PySpark
py_spark_code = read_s3_content('bucket_name', 'large_py_spark.py')
exec(py_spark_code)
```

Résolution des problèmes liés aux sessions

Utilisez les informations de cette rubrique pour résoudre les problèmes liés aux sessions.

Session en mauvais état

Si vous recevez le message d'erreur Session in unhealthy state (Session en mauvais état). Veuillez créer une nouvelle session, mettre fin à votre session existante et en créer une nouvelle.

Impossible d'établir une connexion au serveur du bloc-notes

Lorsque vous ouvrez un bloc-notes, il est possible que le message d'erreur suivant s'affiche :

```
A connection to the notebook server could not be established.
The notebook will continue trying to reconnect.
Check your network connection or notebook server configuration.
```

Cause

Lorsqu'Athena ouvre un bloc-notes, Athena crée une session et se connecte au bloc-notes à l'aide d'une URL de bloc-notes pré-signée. La connexion à l'ordinateur portable utilise le protocole WSS (WebSocketSecure).

L'erreur peut se produire pour les raisons suivantes :

- Un pare-feu local (par exemple, un pare-feu à l'échelle de l'entreprise) bloque le trafic WSS.
- Un proxy ou un logiciel anti-virus sur votre ordinateur local bloque la connexion WSS.

Solution

Supposons que vous ayez une connexion WSS dans la région `us-east-1` comme suit :

```
wss://94c2bcdf-66f9-4d17-9da6-7e7338060183.analytics-gateway.us-east-1.amazonaws.com/
api/kernels/33c78c82-b8d2-4631-bd22-1565dc6ec152/channels?session_id=
7f96a3a048ab4917b6376895ea8d7535
```

Pour résoudre cette erreur, utilisez l'une des stratégies suivantes.

- Utilisez la syntaxe du modèle générique pour autoriser le trafic WSS de liste sur le port `443` à travers Régions AWS et Comptes AWS.

  ```
  wss://*amazonaws.com
  ```

- Utilisez la syntaxe du modèle générique pour autoriser le trafic WSS répertorié sur le port 443 dans un port Région AWS et entre les ports Comptes AWS dans le port Région AWS que vous spécifiez. L'exemple suivant utilise `us-east-1`.

```
wss://*analytics-gateway.us-east-1.amazonaws.com
```

Résolution des problèmes liés aux tables

Impossible de créer une erreur de chemin lors de la création d'une table

Message d'erreur IllegalArgumentException : Impossible de créer un chemin à partir d'une chaîne vide.

Cause : Cette erreur peut se produire lorsque vous utilisez Apache Spark dans Athena pour créer une table dans une AWS Glue base de données et que la propriété de la base de données est vide`LOCATION`.

Solution suggérée : pour plus d'informations et de solutions, voir Exception d'argument non valide lors de la création d'une table.

AccessDeniedException lors de l'interrogation de tables AWS Glue

Message d'erreur : pyspark.sql.utils. AnalysisException: Impossible de vérifier l'existence de la base de données par défaut : com.amazonaws.services.glue.model. AccessDeniedException*: L'utilisateur : arn:aws:sts : ::assumed-role/ aws-account-id-AWSAthenaSparkExecutionRole unique-identifier/- AthenaExecutor unique-identifier n'est pas autorisé à exécuter : glue : on resource GetDatabase : arn:aws:glue : aws-region ::catalog car aws-account-idaucune politique basée sur l'identité n'autorise l'action glue : (Service : ; Code d'état : 400 ; Code d'erreur : ; ID de demande : request-id ; Proxy : null) GetDatabase AWSGlue AccessDeniedException*

Cause : le rôle d'exécution de votre groupe de travail activé par Spark ne dispose pas des autorisations nécessaires pour accéder aux ressources. AWS Glue

Solution suggérée : pour résoudre ce problème, accordez à votre rôle d'exécution l'accès aux AWS Glue ressources, puis modifiez votre politique de compartiment Amazon S3 pour accorder l'accès à votre rôle d'exécution.

La procédure suivante décrit de manière plus détaillée ces étapes.

Pour accorder à votre rôle d'exécution l'accès aux AWS Glue ressources

1. Ouvrez la console Athena à l'adresse https://console.aws.amazon.com/athena/.
2. Si le panneau de navigation de la console n'est pas visible, choisissez le menu d'extension sur la gauche.

3. Dans le panneau de navigation de la console Athena, choisissez Workgroups (Groupes de travail).
4. Sur la page Workgroups (Groupes de travail), choisissez le lien du groupe de travail à consulter.
5. Sur la page Overview Details (Présentation détaillée) du groupe de travail, cliquez sur le lien Role ARN (ARN de rôle). Le lien ouvre le rôle d'exécution Spark dans la console IAM.
6. Dans la section Permissions policies (Politiques d'autorisations), choisissez le nom de la politique de rôle liée.
7. Choisissez Edit policy (Modifier une politique), puis choisissez JSON.
8. Ajoutez AWS Glue l'accès au rôle. En général, vous ajoutez des autorisations pour les actions `glue:GetDatabase` et `glue:GetTable`. Pour plus d'informations sur la configuration des rôles IAM, voir Ajout et suppression d'autorisations basées sur l'identité IAM dans le Guide de l'utilisateur IAM.
9. Choisissez Review policy (Examiner une stratégie), puis Save changes (Enregistrer les modifications).
10. Modifiez la politique de votre compartiment Amazon S3 pour accorder l'accès au rôle d'exécution. Notez que vous devez accorder au rôle l'accès à la fois au compartiment et aux

objets qu'il contient. Pour les étapes, voir Ajout d'une politique de compartiment à l'aide de la console Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

Obtention de support

Pour obtenir de l'aide auprès de AWS, choisissez Support, Support Center dans le AWS Management Console. Pour faciliter votre expérience, veuillez avoir les informations suivantes à portée de main :

- ID de la requête Athena
- ID de session
- ID du calcul

Notes de mise à jour

Décrit les fonctions, les améliorations et les corrections de bogues d'Amazon Athena par date de publication.

Rubriques

Notes de publication d'Athena pour 2024

26 juin 2024

Publié le 2024-06-26

La capacité provisionnée est désormais généralement disponible dans les régions d'Amérique du Sud (São Paulo) et d'Europe (Espagne). La capacité provisionnée vous permet d'exécuter des requêtes SQL sur une capacité de calcul entièrement gérée et fournit des fonctionnalités de gestion des charges de travail qui vous aident à hiérarchiser, contrôler et dimensionner vos charges de travail interactives les plus importantes. Vous pouvez ajouter une capacité à tout moment pour augmenter le nombre de requêtes que vous pouvez exécuter simultanément, contrôler les charges de travail utilisant cette capacité et partager la capacité entre les charges de travail.

Pour plus d’informations, consultez Gestion de la capacité de traitement des requêtes. Pour obtenir des informations sur la tarification, consultez la page Tarification Amazon Athena.

26 avril 2024

Publié le 2024-04-26

Athena publie la version 3.2.0 du pilote JDBC. Pour plus d'informations sur cette version du pilote, consultezNotes de mise à jour d'Amazon Athena JDBC 3.x. Pour télécharger le pilote JDBC 3.x, veuillez consulter Téléchargement du pilote JDBC 3.x.

24 avril 2024

Publié le 24-04-24

Athena annonce les correctifs et améliorations suivants.

- Parquet — Athena prend désormais en charge les lectures rétrocompatibles dans Parquet pour les champs primitifs répétés et non annotés qui ne figurent pas dans une liste ou un groupe de cartes. Cette modification empêche le renvoi de résultats incorrects en silence et améliore les messages d'erreur en cas de non-concordance entre les schémas.

  Pour plus d'informations, consultez la section Support des lectures rétrocompatibles pour les champs primitifs répétés non annotés dans Parquet sur .com. GitHub
- Iceberg OPTIMIZE — Résolution d'un problème lié aux `OPTIMIZE` requêtes qui entraînait la perte de données lorsqu'un filtre autre qu'une clé de partition était utilisé dans une `WHERE` clause. Pour plus d’informations, consultez OPTIMIZE.

16 avril 2024

Publié le 16/04/2024-04

Utilisez la nouvelle fonctionnalité de transmission de requêtes fédérée Amazon Athena pour exécuter des requêtes complètes directement sur la source de données sous-jacente. Les requêtes intermédiaires fédérées vous aident à tirer parti des fonctions uniques, du langage de requête et des capacités de performance de la source de données d'origine. Par exemple, vous pouvez exécuter des requêtes Athena sur DynamoDB à l'aide du langage partiQL. Les requêtes intermédiaires fédérées sont également utiles lorsque vous souhaitez exécuter des `SELECT` requêtes qui regroupent, joignent ou invoquent des fonctions de votre source de données qui ne sont pas disponibles dans Athena. L'utilisation de requêtes directes permet de réduire la quantité de données traitées par Athena et d'accélérer les temps de requête.

Pour plus d’informations, consultez Exécution de requêtes directes fédérées. Pour mettre à niveau les connecteurs que vous utilisez aujourd'hui vers la dernière version, voirMise à jour d'un connecteur de source de données.

10 avril 2024

Publié le 10/04/2024-04

Athena annonce les fonctions et améliorations suivantes.

pilote ODBC 1.2.3.1000

Sortie du pilote ODBC 1.2.3.1000 pour Athena.

Problèmes résolus :

- Problème de connexion au serveur proxy : lorsqu'un serveur proxy était utilisé sans le certificat racine, le connecteur ne parvenait pas à établir de connexion.

Pour plus d'informations et pour télécharger le pilote ODBC 1.x, les notes de version et la documentation, consultez. Pilote ODBC 1.x d'Athena

pilote JDBC 2.1.5

Sortie du pilote JBDC 2.1.5 pour Athena.

Mises à jour et améliorations :

- Mise à jour du SDK AWS Java pour utiliser la version 1.12.687.
- Bibliothèques Jackson mises à jour pour utiliser la version 2.16.0.
- Bibliothèques Logback mises à jour pour utiliser la version 1.3.14.

Pour plus d'informations et pour télécharger le pilote JDBC 2.x, les notes de version et la documentation, consultez. Pilote JDBC 2.x d'Athena

8 avril 2024

Publié le 2024-04-08

Athena annonce la version 2.0.3.0 du pilote ODBC. Pour plus d'informations, consultez les notes de mise à jour de 2.0.3.0. Pour télécharger le nouveau pilote ODBC v2, veuillez consulter Téléchargement du pilote ODBC 2.x. Pour obtenir des informations de connexion, veuillez consulter Amazon Athena ODBC 2.x.

15 mars 2024

Publié le 2024-03-18

Amazon Athena annonce la disponibilité d'Athena SQL dans la région du Canada Ouest (Calgary).

Pour une liste complète des services Services AWS disponibles dans chacun d'entre eux Région AWS, voir AWS Services par région.

15 février 2024

Publié le 15/02/2020

Athena publie la version 3.1.0 du pilote JDBC.

La version 3.1.0 du pilote Amazon Athena JDBC ajoute la prise en charge de l'authentification intégrée Windows Microsoft Active Directory Federation Services (AD FS) et de l'authentification par formulaire. La version 3.1.0 inclut également d'autres améliorations mineures et des corrections de bogues.

Pour télécharger le pilote JDBC v3, consultez. Téléchargement du pilote JDBC 3.x

31 janvier 2024

Publié le 2024-01-31

Athena annonce les fonctions et améliorations suivantes.

- Mise à niveau de Hudi — Vous pouvez désormais utiliser Athena SQL pour interroger les tables Hudi 0.14.0. Pour plus d'informations sur l'utilisation d'Athena SQL pour interroger les tables Hudi, consultez. Utilisation d'Athena pour interroger des jeux de données Apache Hudi

Notes de mise à jour d'Athena pour 2023

14 décembre 2023

Date de publication : 14/12/2023

Athena annonce les correctifs et améliorations suivants.

Athena publie la version 2.1.3 du pilote JDBC. Le pilote résout les problèmes suivants :

- La journalisation a été améliorée pour éviter les conflits avec la journalisation des applications Spring Boot et Gradle.
- Lorsque la méthode JDBC `executeBatch()` était utilisée pour insérer des enregistrements, le pilote n'insérait qu'un seul enregistrement. Athena ne prenant pas en charge l'exécution par lots de requêtes, le pilote signale désormais une erreur lorsque vous utilisez `executeBatch()`. Pour contourner cette limitation, vous pouvez soumettre des requêtes uniques dans une boucle.

Pour télécharger le nouveau pilote JDBC, les notes de mise à jour et la documentation, consultez Pilote JDBC 2.x d'Athena.

9 décembre 2023

Date de publication : 09/12/2023

Publication du pilote ODBC 1.2.1.1000 pour Athena.

Fonctionnalités et améliorations :

- Mise à jour de la prise en charge de RStudio : le pilote ODBC est désormais compatible avec RStudio sur macOS.
- Prise en charge des catalogues et schémas uniques : le connecteur peut désormais renvoyer un catalogue et un schéma uniques. Pour plus d'informations, consultez le guide de configuration et d'installation téléchargeable.

Problèmes résolus :

- Instructions préparées : lorsque des instructions préparées avec un tableau de paramètres à l'aide d'un schéma en colonnes étaient exécutées, le connecteur renvoyait un résultat de requête incorrect.
- Taille de colonne : lorsque la colonne système `$file_modified_time` était sélectionnée, le connecteur renvoyait une taille de colonne incorrecte.
- SQLPrepare : lorsque des paramètres relatifs à `SQLPrepare` dans des requêtes `SELECT` étaient liés, le connecteur renvoyait une erreur.

Pour plus d'informations et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, consultez Pilote ODBC 1.x d'Athena.

7 décembre 2023

Date de publication : 07/12/2023

Athena annonce la version 2.0.2.1 du pilote ODBC. Pour plus d'informations, consultez les notes de mise à jour de 2.0.2.1. Pour télécharger le nouveau pilote ODBC v2, veuillez consulter Téléchargement du pilote ODBC 2.x. Pour obtenir des informations de connexion, veuillez consulter Amazon Athena ODBC 2.x.

5 décembre 2023

Date de publication : 05/12/2023

Vous pouvez désormais créer des groupes de travail Athena SQL utilisant AWS IAM Identity Center le mode d'authentification. Ces groupes de travail prennent en charge la fonctionnalité de propagation d'identité approuvée d'IAM Identity Center. La propagation fiable des identités permet d'utiliser les identités dans des services AWS d'analyse tels qu'Amazon Athena et Amazon EMR Studio.

Pour plus d'informations, consultez Utilisation des groupes de travail Athena compatibles avec IAM Identity Center.

28 novembre 2023

Date de publication : 28/11/2023

Vous pouvez désormais interroger les données dans la classe de stockage Amazon S3 Express One Zone pour des résultats de requête rapides. S3 Express One Zone est une classe de stockage haute performance à zone de disponibilité unique, spécialement conçue pour fournir un accès constant de l'ordre de la milliseconde aux données les plus fréquemment consultées et à celles des applications sensibles à la latence. Pour démarrer, déplacez vos données vers le stockage S3 Express One Zone et cataloguez-les avec AWS Glue Data Catalog pour une expérience de requête fluide dans Athena.

Pour plus d'informations, consultez Interrogation des données de S3 Express One Zone.

27 novembre 2023

Date de publication : 27/11/2023

Athena annonce les fonctions et améliorations suivantes.

- Vues du catalogue de données Glue : les vues du catalogue de données Glue fournissent une vue commune unique sur AWS des services tels qu'Amazon Athena et Amazon Redshift. Dans les

affichages du Catalogue de données Glue, les autorisations d’accès sont définies par l’utilisateur qui a créé l’affichage, et non par l’utilisateur qui interroge l’affichage. Ces affichages permettent de mieux contrôler l’accès, aident à garantir le caractère complet des enregistrements, offrent une sécurité renforcée et peuvent empêcher l’accès aux tables sous-jacentes.

Pour plus d’informations, consultez Utilisation des AWS Glue Data Catalog vues.

- CloudTrail Support Lake — Vous pouvez désormais utiliser Amazon Athena pour analyser les données dans AWS CloudTrail Lake. AWS CloudTrail Lake est un lac de données géré CloudTrail que vous pouvez utiliser pour agréger, stocker et analyser de manière immuable les journaux d'activité à des fins d'audit, de sécurité et d'enquêtes opérationnelles. Pour interroger vos journaux d'activité CloudTrail du lac auprès d'Athena, vous n'avez pas besoin de déplacer des données ou de créer des pipelines de traitement de données distincts. Aucune opération ETL n’est requise.

  Pour commencer, activez la fédération des données dans CloudTrail Lake. Lorsque vous partagez les métadonnées de votre magasin de données d'événements CloudTrail Lake avec AWS Glue Data Catalog, que vous CloudTrail créez les AWS Glue Data Catalog ressources nécessaires et que vous enregistrez les données auprès de AWS Lake Formation. Dans Lake Formation, vous pouvez spécifier les utilisateurs et les rôles qui peuvent utiliser Athena pour interroger votre entrepôt de données d’événements.

  Pour plus d’informations, consultez la rubrique Enable Lake query federation dans le Guide de l’utilisateur AWS CloudTrail .

17 novembre 2023

Date de publication : 17/11/2023

Athena annonce les fonctions et améliorations suivantes.

Fonctionnalités

- Optimiseur basé sur les coûts — Athena annonce la disponibilité générale de l'optimisation basée sur les coûts à l'aide des statistiques de. AWS Glue Pour optimiser vos requêtes dans Athena SQL, vous pouvez demander à Athena de recueillir des statistiques au niveau des tables ou des colonnes pour vos tables dans AWS Glue. Si toutes les tables de votre requête contiennent des statistiques, Athena utilise ces statistiques pour examiner d'autres plans d'exécution et sélectionner celui ayant le plus de chances d'être le plus rapide.

  Pour plus d’informations, consultez Utilisation de l'optimiseur basé sur les coûts.

- Intégration à Amazon EMR Studio : vous pouvez désormais utiliser Athena dans un Amazon EMR Studio sans avoir à utiliser directement la console Athena. Avec l'intégration Athena dans Amazon EMR, vous pouvez effectuer les tâches suivantes :
  - Exécuter des requêtes Athena SQL
  - Afficher les résultats des requêtes
  - Afficher l'historique des requêtes
  - Afficher les requêtes enregistrées
  - Exécuter des requêtes paramétrées
  - Afficher les bases de données, les tables et les vues d'un catalogue de données

  Pour plus d'informations, consultez Amazon EMR Studio dans la rubrique Service AWS intégrations avec Athena.
- Contrôle d'accès imbriqué : Athena annonce la prise en charge du contrôle d'accès aux données imbriquées dans Lake Formation. Dans Lake Formation, vous pouvez définir et appliquer des filtres de données sur des colonnes imbriquées contenant des types de données `struct`. Vous pouvez utiliser le filtrage des données pour restreindre l'accès des utilisateurs aux sous-structures des colonnes imbriquées. Pour plus d'informations sur la création de filtres de données pour les données imbriquées, veuillez consulter Créer un filtre de données dans le Guide du développeur AWS Lake Formation (langue française non garantie).
- Mesures d'utilisation de la capacité allouée — Athena annonce de CloudWatch nouvelles mesures pour les réservations de capacité. Vous pouvez utiliser les nouvelles métriques pour suivre le nombre de DPU que vous avez allouées et le nombre de DPU utilisées par vos requêtes. Lorsque les requêtes sont terminées, vous pouvez également afficher le nombre de DPU consommées par la requête.

  Pour plus d'informations, consultez Surveillance des requêtes Athena à l'aide de métriques CloudWatch .

Améliorations

- Modification du message d'erreur : le message d'erreur `Insufficient Lake Formation permissions` est désormais libellé `Table not found` ou `Schema not found`. Cette modification a été apportée pour empêcher les acteurs malveillants de déduire l'existence de ressources de table ou de base de données à partir du message d'erreur.

16 novembre 2023

Date de publication : 16/11/2023

Athena publie un nouveau pilote JDBC qui améliore l'expérience de connexion, d'interrogation et de visualisation des données à partir d'applications de développement SQL et de business intelligence compatibles. La mise à jour du nouveau pilote est simple. Le pilote peut lire les résultats des requêtes directement à partir d'Amazon S3, ce qui permet de les mettre à votre disposition plus rapidement.

Pour plus d'informations, consultez Pilote Athena JDBC 3.x.

31 octobre 2023

Date de publication : 31/10/2023

Amazon Athena annonce des réserves d'une heure pour la capacité allouée. À compter d'aujourd'hui, vous pouvez réserver et libérer de la capacité allouée au bout d'une heure. Cette modification simplifie l'optimisation des coûts pour les charges de travail dont la demande évolue au fil du temps.

La capacité allouée est une fonctionnalité d'Athena qui fournit des capacités de gestion des charges de travail vous permettant de hiérarchiser, de contrôler et de mettre à l'échelle vos charges de travail interactives les plus importantes. Vous pouvez ajouter une capacité à tout moment pour augmenter le nombre de requêtes que vous pouvez exécuter simultanément, contrôler les charges de travail utilisant cette capacité et partager la capacité entre les charges de travail.

Pour plus d'informations, consultez Gestion de la capacité de traitement des requêtes. Pour obtenir des informations sur la tarification, consultez la page de Tarification d'Amazon Athena.

25 octobre 2023

Date de publication : 26/10/2023

Athena annonce les correctifs et améliorations suivants.

Package jackson-core : le texte JSON dont la valeur numérique est supérieure à 1 000 caractères échouera désormais. Ce correctif résout le problème de sécurité sonatype-2022-6438.

17 octobre 2023

Date de publication : 17/10/2023

Athena annonce la version 2.0.2.0 du pilote ODBC. Pour plus d'informations, consultez les notes de mise à jour de 2.0.2.0. Pour télécharger le nouveau pilote ODBC v2, veuillez consulter Téléchargement du pilote ODBC 2.x. Pour obtenir des informations de connexion, veuillez consulter Amazon Athena ODBC 2.x.

26 septembre 2023

Date de publication : 26/09/2023

Athena annonce les fonctions et améliorations suivantes.

- Support de lecture de Lake Formation pour les tables Delta Lake. Pour plus d'informations sur l'utilisation des tables Delta Lake avec Athena, veuillez consulter Interrogation des tables Linux Foundation Delta Lake.

23 août 2023

Date de publication : 23/08/2023

Amazon Athena annonce la disponibilité d'Athena SQL dans la région d'Israël (Tel Aviv).

Pour une liste complète des services Services AWS disponibles dans chacun d'entre eux Région AWS, voir AWS Services par région.

10 août 2023

Date de publication : 10/08/2023

Athena annonce les correctifs et améliorations suivants.

Version 2.0.1.1 du pilote ODBC

Athena annonce la version 2.0.1.1 du pilote ODBC. Pour plus d'informations, consultez les notes de mise à jour de 2.0.1.1. Pour télécharger le nouveau pilote ODBC v2, veuillez consulter Téléchargement du pilote ODBC 2.x. Pour obtenir des informations de connexion, veuillez consulter Amazon Athena ODBC 2.x.

Version 2.1.1 du pilote JDBC

Athena publie la version 2.1.1 du pilote JDBC. Le pilote résout les problèmes suivants :

- Erreur survenue lors de la création d'une table avec une instruction contenant une expression régulière.
- Problème causant une application incorrecte du paramètre de connexion `ApplicationName`.

Pour télécharger le nouveau pilote JDBC, les notes de mise à jour et la documentation, consultez Connexion à Amazon Athena avec JDBC.

31 juillet 2023

Date de publication : 31/07/2023

Amazon Athena annonce la disponibilité d'Athena SQL dans des Régions AWS supplémentaires.

Cette version étend la disponibilité d'Athena SQL pour inclure l'Asie-Pacifique (Hyderabad), l'Asie-Pacifique (Melbourne), l'Europe (Espagne) et l'Europe (Zurich).

Pour une liste complète des services Services AWS disponibles dans chacun d'entre eux Région AWS, voir AWS Services par région.

27 juillet 2023

Date de publication : 27/07/2023

Athena lance la version 2023.30.1 BigQuery du connecteur Google. Cette version du connecteur réduit le temps d'exécution des requêtes et permet d'effectuer des requêtes sur des points de terminaison BigQuery privés.

Pour plus d'informations sur le BigQuery connecteur Google, consultezConnecteur Amazon Athena pour Google BigQuery . Pour de plus amples informations sur la mise à jour de vos connecteurs de source de données existants, consultez Mise à jour d'un connecteur de source de données.

24 juillet 2023

Date de publication : 24/07/2023

Athena annonce les correctifs et améliorations suivants.

- Requêtes comprenant des unions : amélioration des performances de certaines requêtes comprenant des unions.

- Jointures comprenant des comparaisons de types : correction d'un échec de requête potentiel des instructions `JOIN` incluant une comparaison entre deux types différents.
- Sous-requêtes sur des colonnes imbriquées : correction d'un problème lié aux échecs de requêtes lorsque les sous-requêtes étaient corrélées sur des colonnes imbriquées.
- Vues Iceberg : correction d'un problème de compatibilité lié à la précision des colonnes d'horodatage dans les vues Apache Iceberg. Les vues Iceberg comportant des colonnes d'horodatage sont désormais lisibles, que les colonnes aient été créées sur la version 2 ou 3 du moteur Athena.

20 juillet 2023

Date de publication : 20/07/2023

Athena publie la version 2.1.0 du pilote JDBC. Le pilote inclut de nouvelles améliorations et a résolu un problème.

Améliorations

Les bibliothèques d'analyseurs JSON de Jackson suivantes ont été mises à niveau :

- jackson-annotations 2.15.2 (auparavant 2.14.0)
- jackson-core 2.15.2 (auparavant 2.14.0)
- jackson-databind 2.15.2 (auparavant 2.14.0)

Problèmes résolus

- Correction d'un problème de transfert de paramètres de tableau lors de l'utilisation de la bibliothèque sql2o.

Pour plus d'informations et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, consultez Connexion à Amazon Athena avec JDBC.

13 juillet 2023

Date de publication : 19/09/2023

Athena annonce les fonctions et améliorations suivantes.

- EXPLAIN ANALYZE : ajout de la prise en charge de la file d'attente, de l'analyse, de la planification et de la durée d'exécution à la sortie de `EXPLAIN ANALYZE`.
- EXPLAIN : la sortie `EXPLAIN` affiche désormais des statistiques lorsque la requête contient des agrégations.
- Parquet Hive SerDe — Ajout de la `parquet.ignore.statistics` propriété permettant d'ignorer les statistiques de traitement lors de la lecture des données Parquet. Pour plus d'informations, veuillez consulter Omission des statistiques Parquet.

Pour plus d'informations sur `EXPLAIN` et `EXPLAIN ANALYZE`, consultez Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena. Pour plus d'informations sur le Parquet Hive SerDe, consultezParquet SerDe.

3 juillet 2023

Date de publication : 25/07/2023

Depuis le 3 juillet 2023, Athéna a commencé à supprimer les chaînes de requête des journaux. CloudTrail La chaîne de requête a désormais une valeur de `***OMITTED***`. Cette modification a été apportée pour empêcher la divulgation involontaire de noms de tables ou de valeurs de filtres susceptibles d'inclure des informations sensibles. Si vous utilisiez auparavant les CloudTrail journaux pour accéder aux chaînes de requête complètes, nous vous recommandons d'utiliser l'`Athena::GetQueryExecution`API et de transmettre la valeur de `responseElements.queryExecutionId` from the CloudTrail log. Pour plus d'informations, consultez l'GetQueryExecutionaction dans le manuel Amazon Athena API Reference.

30 juin 2023

Date de publication : 30/06/2023

L'éditeur de requêtes Athena prend désormais en charge les suggestions de code de saisie anticipée pour une expérience de création de requêtes plus rapide. Vous pouvez maintenant écrire des requêtes SQL avec une précision et une efficacité accrues à l'aide des fonctionnalités suivantes :

- Au fur et à mesure que vous tapez, des suggestions apparaissent en temps réel pour les mots-clés, les variables locales, les extraits et les éléments du catalogue.
- Lorsque vous tapez le nom d'une base de données ou d'une table suivi d'un point, l'éditeur affiche facilement une liste de tables ou de colonnes parmi lesquelles choisir.

- Lorsque vous passez le pointeur sur une suggestion d'extrait, un résumé présente un bref aperçu de la syntaxe et de l'utilisation de l'extrait.
- Pour améliorer la lisibilité du code, les mots-clés et leurs règles de mise en surbrillance ont également été mis à jour pour s'aligner sur la dernière syntaxe de Trino et Hive.

Cette caractéristique est activée par défaut. Vous pouvez activer ou désactiver cette fonctionnalité dans les paramètres de préférences de l'éditeur de code.

Pour essayer les suggestions de code de saisie anticipée dans l'éditeur de requêtes Athena, rendez-vous sur la console Athena à l'adresse https://console.aws.amazon.com/athena/.

29 juin 2023

Date de publication : 29/06/2023

- Athena annonce la version 2.0.1.0 du pilote ODBC. Pour plus d'informations, consultez les notes de mise à jour de 2.0.1.0. Pour télécharger le nouveau pilote ODBC v2, veuillez consulter Téléchargement du pilote ODBC 2.x. Pour obtenir des informations de connexion, veuillez consulter Amazon Athena ODBC 2.x.
- Athena et ses fonctionnalités sont désormais disponibles dans la région du Moyen-Orient (EAU). Pour une liste complète des services Services AWS disponibles dans chacun d'entre eux Région AWS, voir AWS Services par région.

28 juin 2023

Date de publication : 28/06/2023

Vous pouvez désormais utiliser Amazon Athena pour interroger des objets restaurés à partir des classes de stockage Amazon S3 S3 Glacier Flexible Retrieval (anciennement Glacier) et S3 Glacier Deep Archive. Vous configurez cette fonctionnalité par table. La fonctionnalité est prise en charge uniquement pour les tables Apache Hive sur la version 3 du moteur Athena.

Pour plus d’informations, consultez Interrogation d'objets Amazon S3 Glacier restaurés.

12 juin 2023

Date de publication : 12/06/2023

Athena annonce les correctifs et améliorations suivants.

- Horodatages de Parquet Reader : ajout de la prise en charge de la lecture des horodatages en tant que `bigint` (millis) pour Parquet Reader. Cette mise à jour fournit une parité avec la prise en charge de la version 2 du moteur Athena.
- EXPLAIN ANALYZE : ajout du temps de lecture physique des entrées aux statistiques de requête et à la sortie de `EXPLAIN ANALYZE`. Pour de plus amples informations sur `EXPLAIN ANALYZE`, consultez Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena.
- INSERT : amélioration des performances de requête sur les tables écrites avec `INSERT`. Pour de plus amples informations sur `INSERT`, consultez INSERT INTO.
- Tables Delta Lake : correction d'un problème lié à `DROP TABLE` sur les tables Delta Lake qui empêchait leur suppression complète en cas de modifications simultanées.

8 juin 2023

Date de publication : 08/06/2023

Amazon Athena pour Apache Spark annonce les nouvelles fonctionnalités suivantes.

- Prise en charge des bibliothèques et configurations Java personnalisées : vous pouvez désormais utiliser vos propres packages Java et une configuration personnalisée pour vos sessions Apache Spark dans Athena. Utilisez les propriétés Spark pour spécifier `.jar` des fichiers, des packages ou toute autre configuration personnalisée avec la console Athena AWS CLI, ou l'API Athena. Pour plus d'informations, consultez Ajout de fichiers JAR et de la configuration personnalisée de Spark.
- Prise en charge des tables Apache Hudi, Apache Iceberg et Delta Lake : Athena pour Spark prend désormais en charge les formats de tables de stockage de lacs de données open source Apache Iceberg, Apache Hudi et Linux Foundation Delta Lake. Pour plus d'informations, consultez Utilisation de formats de table autres que Hive dans Amazon Athena pour Apache Spark et les rubriques individuelles relatives à l'utilisation des tables Apache Iceberg, Apache Hudi et Linux Foundation Delta Lake dans Athena pour Spark.
- Prise en charge du chiffrement pour Apache Spark : dans Athena pour Spark, vous pouvez désormais activer le chiffrement des données en transit entre les nœuds Spark et des données locales au repos stockées sur disque par Spark. Pour activer le chiffrement Spark, vous pouvez utiliser la console Athena AWS CLI, ou l'API Athena. Pour plus d'informations, consultez Activation du chiffrement Apache Spark.

Pour plus d'informations sur Amazon Athena pour Apache Spark, consultez Utilisation d'Apache Spark dans Amazon Athena.

2 juin 2023

Date de publication : 02/06/2023

Vous pouvez désormais supprimer les réservations de capacité dans Athéna et utiliser des AWS CloudFormation modèles pour spécifier les réservations de capacité d'Athéna.

- Suppression de réserves de capacité : vous pouvez désormais supprimer les réserves de capacité annulées dans Athena. La réserve doit être annulée avant de pouvoir être supprimée. La suppression d'une réserve de capacité entraîne la suppression immédiate de la réserve de votre compte. La réserve supprimée ne peut plus être référencée, y compris par son ARN. Pour supprimer une réserve, vous pouvez utiliser la console Athena ou l'API Athena. Pour plus d'informations, consultez Suppression d'une réserve de capacité le guide de l'utilisateur Amazon Athena et le manuel de référence DeleteCapacityReservationdes API Amazon Athena.
- Utiliser AWS CloudFormation des modèles pour les réservations de capacité — Vous pouvez désormais utiliser des AWS CloudFormation modèles pour spécifier les réservations de capacité d'Athena à l'aide de la `AWS::Athena::CapacityReservation` ressource. Pour plus d'informations, consultez AWS: :Athena : : CapacityReservation dans le guide de l'AWS CloudFormation utilisateur.

Pour plus d'informations sur l'utilisation des réserves de capacité pour allouer votre capacité dans Athena, consultez Gestion de la capacité de traitement des requêtes.

25 mai 2023

Date de publication : 25/05/2023

Athena a publié des mises à jour du connecteur de source de données qui améliorent les performances des requêtes fédérées. Les nouvelles optimisations de la poussée vers le bas et le filtrage dynamique permettent d'effectuer davantage d'opérations dans la base de données source plutôt que dans Athena. Ces optimisations réduisent la durée d'exécution des requêtes et la quantité de données analysées. Ces améliorations nécessitent la version 3 du moteur Athena.

Les connecteurs suivants ont été mis à jour :

- Stockage Azure Data Lake
- Azure Synapse
- Cloudera Hive

- Cloudera Impala
- Db2
- DynamoDB
- Google BigQuery
- Hortonworks
- MySQL
- Oracle
- PostgreSQL
- Redshift
- SAP HANA
- Snowflake
- SQL Server
- Teradata

Pour de plus amples informations sur la mise à niveau des connecteurs de source de données, consultez Mise à jour d'un connecteur de source de données.

18 mai 2023

Date de publication : 18/05/2023

Vous pouvez désormais utiliser AWS PrivateLink les connexions entrantes IPv6 vers Amazon Athena.

Amazon Athena a étendu sa prise en charge des connexions entrantes via les points de terminaison IPv6 (Internet Protocol version 6) pour y inclure AWS PrivateLink. À compter d'aujourd'hui, vous pouvez vous connecter à Athena de manière sécurisée et privée AWS PrivateLink depuis votre Amazon Virtual Private Cloud (Amazon VPC), en plus des points de terminaison IPv6 publics qui étaient auparavant disponibles.

La croissance rapide d'Internet épuise la disponibilité des adresses IPv4 (Internet Protocol version 4). IPv6 multiplie plusieurs fois le nombre d'adresses disponibles, de sorte que vous n'avez plus à gérer les espaces d'adresses qui se chevauchent dans vos VPC. Avec cette version, vous pouvez désormais combiner les avantages de l'adressage IPv6 avec les avantages de sécurité et de performances de AWS PrivateLink.

Pour vous connecter par programmation à un AWS service, vous pouvez utiliser le AWS SDK AWS CLIou pour spécifier un point de terminaison. Pour plus d'informations sur les points de terminaison de service et les points de terminaison de service Athena, consultez Points de terminaison de service AWS et Points de terminaison et quotas Amazon Athena dans le Référence générale d'Amazon Web Services.

15 mai 2023

Date de publication : 15/05/2023

Athena annonce la sortie des connecteurs Apache Spark DataSource V2 (DSV2) pour DynamoDB, Logs, CloudWatch Metrics et CMDB. CloudWatch AWS Utilisez les nouveaux connecteurs DSV2 pour interroger ces sources de données à l'aide de Spark. Les connecteurs DSV2 utilisent les mêmes paramètres que les connecteurs fédérés Athena correspondants. Les connecteurs DSV2 s'exécutent directement sur les applications de travail Spark et vous n'avez pas besoin de déployer une fonction Lambda pour les utiliser.

Pour plus d’informations, consultez Connecteurs de source de données Athena pour Apache Spark.

10 mai 2023

Date de publication : 10/05/2023

Publication du pilote ODBC 1.1.20 pour Athena.

Fonctionnalités et améliorations :

- Prise en charge du remplacement des point de terminaison Lake Formation.
- Le plug-in d'authentification ADFS dispose d'un nouveau paramètre permettant de définir la valeur de partie utilisatrice (`LoginToRP`).
- AWS mises à jour de la bibliothèque.

Correctifs de bogue :

- Échec de l'annulation de l'allocation de l'instruction préparée lorsque la méthode `SQLPrepare()` n'a pas été soumise.
- Erreur de liaison des paramètres de l'instruction préparée lors de la conversion d'un type C en type SQL.

- Impossible de renvoyer les données quand les requêtes `EXPLAIN` et `EXPLAIN ANALYZE` utilisaient `SQLPrepare()` et`SQLExecute()`.

Pour plus d'informations et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, consultez Connexion à Amazon Athena avec le pilote ODBC.

8 mai 2023

Date de publication : 08/05/2023

Athena annonce les correctifs et améliorations suivants.

- Intégration à Hudi mise à jour : Athena a mis à jour son intégration à Apache Hudi. Vous pouvez désormais utiliser Athena pour interroger les tables Hudi 0.12.2 et le listage des métadonnées Hudi pour les tables Hudi est désormais pris en charge. Pour plus d'informations, consultez Utilisation d'Athena pour interroger des jeux de données Apache Hudi et Listage des métadonnées Hudi.
- Correctif de conversion d'horodatage : correction de la gestion des conversions d'horodatage vers un type de données de moindre précision. Auparavant, la version 3 du moteur Athena arrondissait incorrectement la valeur au type de cible au lieu de la tronquer lors de la conversion.

  Les exemples suivants illustrent la gestion incorrecte avant le correctif.

  Exemple 1 : conversion d'un horodatage en microsecondes en millisecondes

  Exemples de données

```
A, 2020-06-10 15:55:23.383
B, 2020-06-10 15:55:23.382
C, 2020-06-10 15:55:23.383345
D, 2020-06-10 15:55:23.383945
E, 2020-06-10 15:55:23.383345734
F, 2020-06-10 15:55:23.383945278
```

  La requête suivante tente de récupérer les horodatages correspondant à une valeur spécifique.

```
SELECT *
FROM table
WHERE timestamps.col = timestamp'2020-06-10 15:55:23.383'
```

  La requête renvoyait les résultats suivants.

```
A, 2020-06-10 15:55:23.383
C, 2020-06-10 15:55:23.383
E, 2020-06-10 15:55:23.383
```

Avant le correctif, Athena n'incluait pas les valeurs `2020-06-10 15:55:23.383945` ou `2020-06-10 15:55:23.383945278` parce qu'elles avaient été arrondies à `2020-06-10 15:55:23.384`.

Exemple 2 : conversion d'un horodatage en date

La requête suivante renvoyait un résultat erroné.

```
SELECT date(timestamp '2020-12-31 23:59:59.999')
```

Résultat

`2021-01-01`

Avant le correctif, Athena arrondissait la valeur, avançant ainsi la journée. Ces valeurs sont désormais tronquées au lieu d'être arrondies.

28 avril 2023

Date de publication : 28/04/2023

Vous pouvez désormais utiliser les réserves de capacité sur Amazon Athena pour exécuter des requêtes SQL sur une capacité de calcul entièrement gérée.

la capacité allouée fournit des capacités de gestion des charges de travail qui vous aident à hiérarchiser, contrôler et mettre à l'échelle vos charges de travail interactives les plus importantes. Vous pouvez ajouter une capacité à tout moment pour augmenter le nombre de requêtes que vous pouvez exécuter simultanément, contrôler les charges de travail utilisant cette capacité et partager la capacité entre les charges de travail.

Pour plus d’informations, consultez Gestion de la capacité de traitement des requêtes. Pour obtenir des informations sur la tarification, consultez la page Tarification Amazon Athena.

17 avril 2023

Date de publication : 17/04/2023

Athena publie la version 2.0.36 du pilote JDBC. Le pilote inclut de nouvelles fonctionnalités et a résolu un problème.

Nouvelles fonctionnalités

- Vous pouvez désormais utiliser des identifiants de parties utilisatrices personnalisables avec l'authentification AD FS.
- Vous pouvez désormais ajouter le nom de l'application qui utilise le connecteur à la chaîne de l'agent utilisateur.

Problèmes résolus

- Correction d'une erreur qui se produisait lors de l'utilisation de `getSchema()` pour récupérer un schéma inexistant.

Pour plus d'informations et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, consultez Connexion à Amazon Athena avec JDBC.

14 avril 2023

Date de publication : 20/06/2023

Athena annonce les correctifs et améliorations suivants.

- Lorsque vous convertissez une chaîne en horodatage, un espace est requis entre le jour et l'heure ou le fuseau horaire. Pour plus d’informations, consultez Espace requis entre les valeurs de date et d'heure lors de la conversion d'une chaîne en un horodatage.
- Suppression d'un changement critique dans la façon dont la précision de l'horodatage était gérée. Pour garantir la cohérence entre les versions 2 et 3 du moteur Athena, la précision de l'horodatage est désormais définie par défaut en millisecondes au lieu de microsecondes.
- Athena impose désormais systématiquement l'accès au compartiment de sortie des requêtes lorsqu'elle exécute des requêtes. Assurez-vous que tous les principaux IAM qui exécutent l'StartQueryExecutionaction disposent de l'GetBucketLocationautorisation S3 : sur le compartiment de sortie de la requête.

4 avril 2023

Date de publication : 04/04/2023

Vous pouvez désormais utiliser Amazon Athena pour créer et interroger des vues sur des sources de données fédérées. Utilisez une vue fédérée unique pour interroger plusieurs tables externes ou sous-jeux de données. Cela simplifie le SQL requis et vous permet d'obscurcir les sources de données des utilisateurs finaux qui doivent utiliser le SQL pour interroger les données.

Pour plus d'informations, consultez Utilisation des vues et Exécution de requêtes fédérées.

30 mars 2023

Date de publication : 30/03/2023

Amazon Athena annonce la disponibilité d'Amazon Athena pour Apache Spark dans des Régions AWS supplémentaires.

Cette version étend la disponibilité d'Amazon Athena pour Apache Spark pour inclure l'Asie-Pacifique (Mumbai), l'Asie-Pacifique (Singapour), l'Asie-Pacifique (Sydney) et l'Europe (Francfort).

Pour plus d'informations sur Amazon Athena pour Apache Spark, consultez Utilisation d'Apache Spark dans Amazon Athena.

28 mars 2023

Date de publication : 28/03/2023

Athena annonce les correctifs et améliorations suivants.

- Dans les réponses aux actions d'API Athena `GetQueryExecution` et `BatchGetQueryExecution`, le nouveau champ `subStatementType` indique le type de requête exécutée (par exemple, `SELECT`, `INSERT`, `UNLOAD`, `CREATE_TABLE` ou `CREATE_TABLE_AS_SELECT`).
- Correction d'un bogue qui entraînait un chiffrement incorrect des fichiers manifestes pour les opérations d'écriture d'Apache Hive.
- La version 3 du moteur Athena gère désormais correctement les valeurs `NaN` et `Infinity` dans la fonction `approx_percentile`. La fonction `approx_percentile` renvoie le percentile approximatif d'un jeu de données au pourcentage donné.

La version 2 du moteur Athena traite incorrectement `NaN` comme une valeur supérieure à `Infinity`. La version 3 du moteur Athena gère désormais `NaN` et `Infinity` conformément au traitement de ces valeurs dans d'autres fonctions analytiques et statistiques. Les points suivants décrivent le nouveau comportement de manière plus détaillée.

- Si `NaN` est présent dans le jeu de données, Athena renvoie `NaN`.
- Si `NaN` n'est pas présente, mais que `Infinity` est présent, Athena traite `Infinity` comme un très grand nombre.
- Si plusieurs valeurs `Infinity` sont présentes, Athena les traite comme le même très grand nombre. Si nécessaire, Athena renvoie `Infinity`.
- Si un seul jeu de données contient les deux - `Infinity` et `-Double.MAX_VALUE` - et qu'un résultat en percentile est `-Double.MAX_VALUE`, Athena renvoie `-Infinity`.
- Si un seul jeu de données contient les deux - `Infinity` et `Double.MAX_VALUE` - et qu'un résultat en percentile est `Double.MAX_VALUE`, Athena renvoie `Infinity`.
- Pour exclure `Infinity` et `NaN` d'un calcul, utilisez la fonction `is_finite()`, comme dans l'exemple suivant.

```
approx_percentile(x, 0.5) FILTER (WHERE is_finite(x))
```

27 mars 2023

Date de publication : 27/03/2023

Vous pouvez désormais spécifier un niveau de chiffrement minimal au niveau des groupes de travail Athena SQL dans Amazon Athena. Cette fonctionnalité garantit le chiffrement des résultats de toutes les requêtes du groupe de travail Athena SQL au niveau de chiffrement que vous spécifiez ou supérieur. Vous pouvez choisir entre plusieurs niveaux de puissance de chiffrement pour protéger vos données. Pour configurer le niveau de chiffrement minimal que vous souhaitez, vous pouvez utiliser la console AWS CLI, l'API ou le SDK Athena.

La fonctionnalité de chiffrement minimum n'est pas disponible pour les groupes de travail compatibles avec Apache Spark. Pour plus d’informations, consultez Configuration du chiffrement minimal pour un groupe de travail.

17 mars 2023

Date de publication : 17/03/2023

Athena annonce les correctifs et améliorations suivants.

- Correction d'un problème lié au connecteur Amazon Athena DynamoDB en raison duquel les requêtes échouaient et le message d'erreur ne KeyConditionExpressions devait contenir qu'une seule condition par clé.

  Ce problème se produit car la version 3 du moteur Athena reconnaît la possibilité de pousser vers le bas davantage de types de prédicats que la version 2 du moteur Athena. Dans la version 3 du moteur Athena, des clauses telles que `some_column LIKE 'someprefix%` sont poussées vers le bas sous forme de prédicats de filtre qui appliquent des limites inférieure et supérieure à une colonne donnée. La version 2 du moteur Athena n'a pas poussé ces prédicats vers le bas. Dans la version 3 du moteur Athena, lorsque `some_column` est une colonne de clé de tri, le moteur pousse le prédicat du filtre vers le connecteur DynamoDB. Le prédicat de filtre est ensuite redirigé vers le service DynamoDB. DynamoDB ne prenant en charge qu'une seule condition de filtre sur une clé de tri, DynamoDB renvoie l'erreur.

  Pour résoudre ce problème, mettez à jour votre connecteur Amazon Athena DynamoDB vers la version 2023.11.1. Pour obtenir des instructions sur la mise à jour du connecteur, consultez Mise à jour d'un connecteur de source de données.

8 mars 2023

Date de publication : 08/03/2023

Athena annonce les correctifs et améliorations suivants.

- Correction d'un problème lié aux requêtes fédérées qui entraînait l'envoi des valeurs des prédicats d'horodatage sous forme de microsecondes au lieu de millisecondes.

15 février 2023

Date de publication : 15/02/2023

Athena annonce les correctifs et améliorations suivants.

- Vous pouvez désormais utiliser le chiffrement côté client afin de chiffrer les données dans Amazon S3 pour les opérations d'écriture d'Iceberg.

- Correction d'un problème qui affectait le chiffrement côté serveur dans Amazon S3 pour les opérations d'écriture d'Iceberg.

31 janvier 2023

Date de publication : 31/01/2023

Vous pouvez désormais utiliser Amazon Athena pour interroger les données dans Google Cloud Storage. Comme Amazon S3, Google Cloud Storage est un service géré qui stocke les données dans des compartiments. Utilisez le connecteur Athena pour Google Cloud Storage pour exécuter des requêtes fédérées interactives sur vos données externes.

Pour plus d'informations, consultez Connecteur Amazon Athena Google Cloud Storage.

20 janvier 2023

Date de publication : 20/01/2023

Vous pouvez désormais consulter une documentation complète sur la prise en charge de la compression Athena. Des rubriques individuelles ont été ajoutées pour Compression de la table Hive compression de la table Iceberg, et Niveaux de compression ZSTD.

Pour plus d'informations, consultez Prise en charge de la compression Athena.

3 janvier 2023

Date de publication : 03/01/2023

Athena annonce les mises à jour suivantes :

- Commandes supplémentaires pour les métastores Hive – Vous pouvez utiliser Athena pour vous connecter à votre métastore Apache Hive autogéré en tant que catalogue de métadonnées et interroger des données stockées dans Amazon S3. Dans cette version, vous pouvez utiliser `CREATE TABLE AS` (CTAS), `INSERT INTO` et 12 commandes supplémentaires du langage de définition de données (DDL) pour interagir avec le métastore Apache Hive. Vous pouvez gérer vos schémas de métastore Hive directement à partir d'Athena en utilisant cet ensemble étendu de fonctionnalités SQL.

  Pour plus d'informations, consultez Utilisation du connecteur de données Athena pour un métastore Hive externe.

- Version 2.0.35 du pilote JDBC – Athena publie la version 2.0.35 du pilote JDBC. Le pilote JDBC 2.0.35 contient les mises à jour suivantes :
  - Le pilote utilise maintenant les bibliothèques suivantes pour l'analyseur JSON de Jackson.
    - jackson-annotations 2.14.0 (auparavant 2.13.2)
    - jackson-core 2.14.0 (auparavant 2.13.2)
    - jackson-databind 2.14.0 (auparavant 2.13.2.2)
  - La prise en charge de la version 4.1 de JDBC est interrompue.

  Pour plus d'informations et pour télécharger le nouveau pilote, les notes de mise à jour et la documentation, voir Connexion à Amazon Athena avec JDBC.

Notes de publication d'Athena pour 2022

14 décembre 2022

Date de publication : 14/12/2022

Vous pouvez désormais utiliser le connecteur Amazon Athena pour Kafka pour exécuter des requêtes SQL sur des données en streaming. Par exemple, vous pouvez exécuter des requêtes analytiques sur des données en streaming et en temps réel dans Amazon Managed Streaming for Apache Kafka (Amazon MSK) et les associer aux données historiques de votre lac de données dans Amazon S3.

Le connecteur Amazon Athena pour Kafka prend en charge les requêtes sur plusieurs moteurs de streaming. Vous pouvez utiliser Athena pour exécuter des requêtes SQL sur des clusters provisionnés et sans serveur Amazon MSK, sur des déploiements Kafka autogérés et sur des données en streaming dans Confluent Cloud.

Pour plus d’informations, consultez Connecteur Amazon Athena pour MSK.

2 décembre 2022

Date de publication : 02/12/2022

Athena publie la version 2.0.34 du pilote JDBC. Le pilote JDBC 2.0.34 inclut les nouvelles fonctions suivantes et a résolu les problèmes suivants :

- Prise en charge de la réutilisation des résultats des requêtes – Vous pouvez désormais réutiliser les résultats de requêtes exécutées précédemment jusqu'à une limite de temps que vous spécifiez, au lieu de demander à Athena de recalculer les résultats à chaque exécution de la requête. Pour plus d'informations, consultez le guide d'installation et de configuration, disponible sur la page de téléchargement de JDBC, et Réutilisation des résultats des requêtes.
- InstanceMetadata Support Ec2 — Le pilote JDBC prend désormais en charge la méthode d'InstanceMetadataauthentification Ec2 à l'aide de profils d'instance IAM.
- Correction d'une exception basée sur les caractères – Correction d'une exception qui se produisait avec les requêtes contenant certains caractères linguistiques.
- Correction de vulnérabilité — Correction d'une vulnérabilité liée aux AWS dépendances fournies avec le connecteur.

Pour plus d'informations et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, consultez Connexion à Amazon Athena avec JDBC.

30 novembre 2022

Date de publication : 30/11/2022

Vous pouvez désormais créer et exécuter de manière interactive des applications Apache Spark et des blocs-notes compatibles Jupyter sur Athena. Exécutez des analyses de données sur Athena à l'aide de Spark sans avoir à planifier, configurer ou gérer les ressources. Soumettez le code Spark pour traitement et recevez directement les résultats. Utilisez l'expérience simplifiée du bloc-notes dans la console Amazon Athena pour développer des applications Apache Spark en utilisant Python ou API de bloc-notes Athena.

Apache Spark fonctionne sur Amazon Athena sans serveur et offre une mise à l'échelle automatique et à la demande qui permet d'obtenir un calcul instantané pour répondre à l'évolution des volumes de données et des exigences de traitement.

Pour plus d’informations, consultez Utilisation d'Apache Spark dans Amazon Athena.

18 novembre 2022

Date de publication : 18/11/2022

Vous pouvez désormais utiliser le connecteur Amazon Athena pour Db2 IBM pour interroger Db2 depuis Athena. Par exemple, vous pouvez exécuter des requêtes analytiques sur un entrepôt des données sur Db2 et un lac de données sur Amazon S3.

Le connecteur Db2 d'Amazon Athena expose plusieurs options de configuration par le biais de variables d'environnement Lambda. Pour plus d'informations sur les options de configuration, les paramètres, les chaînes de connexion, le déploiement et les limitations, voir Connecteur Amazon Athena pour Db2 IBM.

17 novembre 2022

Date de publication : 17/11/2022

La prise en charge d'Apache Iceberg dans la version 3 du moteur Athena offre désormais les fonctionnalités de transaction ACID améliorées suivantes :

- Prise en charge d'ORC et d'Avro – Créez des tables Iceberg en utilisant les formats de fichiers basés sur les lignes et les colonnes Apache Avro et Apache ORC. La prise en charge de ces formats s'ajoute à la prise en charge existante de Parquet.
- MERGE INTO – Utilisez la commande `MERGE INTO` pour fusionner efficacement des données à grande échelle. `MERGE INTO` combine les opérations `INSERT`, `UPDATE` et `DELETE` en une seule transaction. Cela réduit la charge de traitement dans votre pipeline de données et nécessite moins de SQL pour l'écriture. Pour plus d'informations, consultez Mise à jour des données de la table Iceberg et MERGE INTO.
- Prise en charge de CTAS et de VIEW – Utilisez les instructions `CREATE TABLE AS SELECT` (CTAS) and `CREATE VIEW` avec les tables Iceberg. Pour plus d'informations, consultez CREATE TABLE AS et CREATE VIEW.
- Prise en charge de VACUUM – Vous pouvez utiliser l'instruction `VACUUM` pour optimiser votre lac de données en supprimant les instantanés et les données qui ne sont plus nécessaires. Vous pouvez utiliser cette fonctionnalité pour améliorer les performances de lecture et répondre aux exigences réglementaires telles que le RGPD. Pour plus d'informations, consultez Optimisation des tables Iceberg et VACUUM.

Ces nouvelles fonctionnalités nécessitent la version 3 du moteur Athena et sont disponibles dans toutes les régions où le service Athena est pris en charge. Vous pouvez les utiliser avec la console Athena, les pilotes ou l'API.

Pour plus d'informations sur l'utilisation d'Iceberg dans Athena, voir Utilisation des tables Apache Iceberg.

14 novembre 2022

Date de publication : 14/11/2022

Amazon Athena prend désormais en charge les points de terminaison IPv6 pour les connexions entrantes que vous pouvez utiliser pour invoquer des fonctions Athena via IPv6. Vous pouvez utiliser cette fonctionnalité pour répondre aux exigences de conformité IPv6. Elle élimine également le besoin d'équipements réseau supplémentaires pour gérer la traduction d'adresses entre IPv4 et IPv6.

Pour utiliser cette fonctionnalité, configurez vos applications afin d'utiliser les nouveaux points de terminaison à double pile d'Athena, qui prennent en charge à la fois IPv4 et IPv6. Les points de terminaison à double pile utilisent le format `athena.region.api.aws`. Par exemple, le point de terminaison à double pile dans la région USA Est (Virginie du Nord) est `athena.us-east-1.api.aws`.

Lorsque vous adressez une requête à un point de terminaison à double pile d'Athena, celui-ci se résout en une adresse IPv6 ou IPv4, selon le protocole utilisé par votre réseau et votre client. Pour vous connecter par programmation à un AWS service, vous pouvez utiliser le AWS SDK AWS CLIou pour spécifier un point de terminaison.

Pour en savoir plus sur les points de terminaison du service, voir points de terminaison de service AWS. Pour en savoir plus sur les points de terminaison du service Athena, voir Points de terminaison et quotas d'Amazon Athena dans la documentation AWS .

Vous pouvez utiliser les nouveaux points de terminaison à double pile d'Athena pour les connexions entrantes sans coût supplémentaire. Les points de terminaison à double pile sont généralement disponibles dans toutes les Régions AWS.

11 novembre 2022

Date de publication : 11/11/2022

Athena annonce les correctifs et améliorations suivants.

- Contrôle d'accès précis Lake Formation étendu – Vous pouvez désormais utiliser des politiques de contrôle d'accès précis AWS Lake Formation dans les requêtes Athena pour les données stockées dans n'importe quel format de fichier ou de table pris en charge. Vous pouvez utiliser un contrôle d'accès précis dans Lake Formation pour restreindre l'accès aux données des résultats des requêtes à l'aide de filtres de données afin de garantir la sécurité au niveau des colonnes, des lignes et des cellules. Les formats de table pris en charge par Athena sont Apache Iceberg,

Apache Hudi et Apache Hive. Le contrôle d'accès précis étendu est disponible dans toutes les régions prises en charge par Athena. La prise en charge étendue des formats de table et de fichier nécessite Version 3 du moteur Athena, qui offre de nouvelles fonctionnalités et améliore les performances des requêtes, mais ne change pas la façon dont vous configurez les politiques de contrôle d'accès précis dans Lake Formation.

L'utilisation de ce contrôle d'accès précis étendu dans Athena a les implications suivantes :

- EXPLAIN – Les informations de filtrage des lignes ou des cellules définies dans Lake Formation et les informations sur les statistiques des requêtes n'apparaissent pas dans la sortie de `EXPLAIN` et `EXPLAIN ANALYZE`. Pour plus d'informations sur `EXPLAIN` dans Athena, voir Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena.
- Métastores Hive externes – Les colonnes cachées d'Apache Hive ne peuvent pas être utilisées pour le filtrage du contrôle d'accès précis, et les tables système cachées d'Apache Hive ne sont pas prises en charge par le contrôle d'accès précis. Pour plus d'informations, consultez Considérations et restrictions dans la rubrique Utilisation du connecteur de données Athena pour un métastore Hive externe.
- Statistiques des requêtes – Les informations relatives au nombre de lignes d'entrée et de sortie et à la taille des données ne figurent pas dans les statistiques des requêtes Athena lorsque des filtres de niveau ligne sont définis dans Lake Formation. Pour plus d'informations sur l'affichage des statistiques relatives aux requêtes Athena, reportez-vous Affichage des statistiques et des détails d'exécution pour les requêtes terminées aux sections et. GetQueryRuntimeStatistics
- Groupes de travail – Les utilisateurs du même groupe de travail Athena peuvent voir les données que le contrôle d'accès précis de Lake Formation a configurées pour être accessibles au groupe de travail. Pour plus d'informations sur l'utilisation d'Athena pour interroger des données enregistrées dans Lake Formation,voir Utilisation d'Athena pour interroger des données enregistrées dans AWS Lake Formation.

Pour en savoir plus sur l'utilisation du contrôle d'accès précis dans Lake Formation, voir Gérer le contrôle d'accès précis à l'aide de AWS Lake Formation sur le blog AWS Big Data.

- Requête fédérée Athena – La requête fédérée d'Athena préserve désormais la casse originale des noms de champs dans les objets `struct`. Auparavant, les noms des champs `struct` étaient automatiquement mis en minuscules.

8 novembre 2022

Date de publication : 08/11/2022

Vous pouvez désormais utiliser la fonction de mise en cache de la réutilisation des résultats des requêtes pour accélérer les requêtes répétées dans Athena. Une requête répétée est une requête SQL identique à une requête soumise récemment et qui produit les mêmes résultats. Lorsque vous devez exécuter plusieurs requêtes identiques, la mise en cache en vue de la réutilisation des résultats peut réduire le temps nécessaire à la production des résultats. La mise en cache en vue de la réutilisation des résultats permet également de réduire les coûts en diminuant le nombre d'octets analysés.

Pour plus d'informations, consultez Réutilisation des résultats des requêtes.

13 octobre 2022

Date de publication : 13/10/2022

Athena annonce la version 3 du moteur Athena.

Athena a mis à jour son moteur de requêtes SQL afin d'inclure les dernières fonctionnalités du projet open source Trino. En plus de prendre en charge toutes les fonctionnalités de la version 2 du moteur Athena, la version 3 inclut plus de 50 nouvelles fonctions SQL, 30 nouvelles fonctionnalités et plus de 90 améliorations des performances des requêtes. Avec le lancement d'aujourd'hui, Athena introduit également une approche d'intégration continue de la gestion des logiciels open source. Celle-ci améliore l'actualité des projets Trino et Presto, afin que vous puissiez accéder plus rapidement aux améliorations de la communauté, intégrées et ajustées au sein du moteur Athena.

Pour plus d'informations, consultez Version 3 du moteur Athena.

10 octobre 2022

Date de publication : 10/10/2022

Athena publie le pilote JDBC version 2.0.33. Le pilote JDBC 2.0.33 comprend les modifications suivantes :

- La nouvelle version du pilote, la version JDBC et les propriétés du nom du plug-in ont été ajoutées à la chaîne de l'agent utilisateur dans la classe du fournisseur d'informations d'identification.
- Les messages d'erreur ont été corrigés et les informations nécessaires ajoutées.
- Les instructions préparées sont désormais désallouées si la connexion est fermée ou si l'exécution d'instruction préparée par Athena échoue.

Pour plus d'informations et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, consultez Connexion à Amazon Athena avec JDBC.

23 septembre 2022

Date de publication : 26/09/2022

Le connecteur Amazon Athena Neptune autorise désormais la mise en correspondance non sensible à la casse pour les noms de colonnes et de tables.

- Le connecteur de source de données Neptune peut résoudre les noms de colonnes sur les tables Neptune qui utilisent la casse, même si les noms des colonnes sont tous en minuscules dans la table de AWS Glue. Pour activer ce comportement, définissez la variable d'environnement `enable_caseinsensitivematch` sur `true` dans la fonction Lambda du connecteur Neptune.
- Étant donné que seuls AWS Glue les noms de table en minuscules sont pris en charge, lorsque vous créez une AWS Glue table pour Neptune, spécifiez le paramètre de AWS Glue table. `"glabel" = table_name`

Pour plus d'informations sur le connecteur Neptune, veuillez consulter la rubrique Connecteur Amazon Athena pour Neptune.

13 septembre 2022

Date de publication : 13/09/2022

Athena annonce les correctifs et améliorations suivants.

- Metastore Hive externe – Athena renvoie maintenant `NULL` au lieu de lancer une exception lorsqu'une clause `WHERE` inclut une partition qui n'existe pas dans un metastore Hive externe (EHMS). Le nouveau comportement correspond à celui du AWS Glue Data Catalog.
- Requêtes paramétrées – Les valeurs dans les requêtes paramétrées peuvent désormais être envoyées au type de données `DOUBLE`.
- Apache Iceberg – Les opérations d'écriture sur des tables Iceberg aboutissent désormais lorsque le verrouillage d'objet est activé sur un compartiment Amazon S3.

31 août 2022

Date de publication : 31/08/2022

Amazon Athena annonce la disponibilité d'Athena et ses fonctions dans la région Asie-Pacifique (Jakarta).

Cette version étend la disponibilité d'Athena dans la région Asie-Pacifique pour inclure Asie-Pacifique (Hong Kong), Asie-Pacifique (Jakarta), Asie-Pacifique (Mumbai), Asie-Pacifique (Osaka), Asie-Pacifique (Séoul), Asie-Pacifique (Singapour), Asie-Pacifique (Sydney) et Asie-Pacifique (Sydney) et Asie-Pacifique (Tokyo). Pour accéder à une liste complète des Services AWS disponibles dans ces régions et dans d'autres, consultez la Région AWS Liste des services régionaux.

23 août 2022

Date de publication : 23/08/2022

La version v2022.32.1 du kit SDK Athena Query Federation comprend les modifications suivantes :

- Ajout de la prise en charge du connecteur de source de données Oracle d'Amazon Athena pour les connexions basées sur SSL aux instances Amazon RDS. La prise en charge est limitée au protocole TLS (Transport Layer Security) et à l'authentification du serveur par le client. Comme l'authentification mutuelle n'est pas prise en charge dans Amazon RDS, la mise à jour n'inclut pas la prise en charge de l'authentification mutuelle.

  Pour plus d'informations, consultez Connecteur Amazon Athena pour Oracle.

3 août 2022

Date de publication : 03/08/2022

Athena publie le pilote JDBC version 2.0.32. Le pilote JDBC 2.0.32 comprend les modifications suivantes :

- La chaîne `User-Agent` envoyée au kit SDK Athena a été étendue pour contenir la version du pilote, la version de spécification JDBC et le nom du plugin d'authentification.
- Correction d'un `NullPointerException` qui était lancé lorsqu'aucune valeur n'était fournie pour le paramètre `CheckNonProxyHost`.
- Correction d'un problème d'`login_url`analyse dans le plugin BrowserSaml d'authentification.
- Correction d'un problème d'hôte proxy qui survenait lorsque le paramètre `UseProxyforIdp` était défini sur `true`.

Pour plus d'informations et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, consultez Connexion à Amazon Athena avec JDBC.

1er août 2022

Date de publication : 01/08/2022

Athena annonce des améliorations apportées au kit SDK Athena Query Federation et aux connecteurs de source de données prédéfinis Athena Les améliorations apportées sont les suivantes :

- Analyse syntaxique des structures – Correction d'un problème d'analyse syntaxique `GlueFieldLexer` dans le kit SDK Athena Query Federation qui empêchait l'affichage de toutes les données de certaines structures complexes. Ce problème a affecté les connecteurs créés sur le kit SDK Athena Query Federation.
- AWS Glue tables — Ajout d'un support supplémentaire pour les types `set` et `decimal` colonnes dans AWS Glue les tableaux.
- Connecteur DynamoDB – Ajout de la possibilité d'ignorer la casse des noms d'attributs DynamoDB. Pour plus d'informations, voir `disable_projection_and_casing` dans la section Paramètres de la page Connecteur Amazon Athena pour DynamoDB.

Pour plus d'informations, consultez la version v2022.30.2 d'Athena Query Federation sur. GitHub

21 juillet 2022

Date de publication : 21/07/2022

Vous pouvez désormais analyser et déboguer vos requêtes à l'aide de mesures de performances et d'outils d'analyse de requêtes visuels interactifs dans la console Athena. Les données de performance des requêtes et les détails d'exécution peuvent vous aider à identifier les goulots d'étranglement dans les requêtes, à inspecter les opérateurs et les statistiques pour chaque étape d'une requête, à suivre le volume de données circulant entre les étapes et à valider l'impact des prédicats de requête. Vous pouvez désormais :

- Accédez au plan d'exécution distribué et logique de votre requête en un seul clic.
- Explorez les opérations à chaque étape avant que l'étape ne soit exécutée.
- Visualisez les performances des requêtes terminées avec des mesures du temps passé dans les étapes de mise en file d'attente, de planification et d'exécution.

- Obtenez des informations sur le nombre de lignes et la quantité de données sources traitées et sorties par votre requête.
- Consultez les détails d'exécution granulaires de vos requêtes, présentés dans leur contexte et formatés sous forme de graphique interactif.
- Utilisez des détails d'exécution précis au niveau de l'étape pour comprendre le flux de données dans votre requête.
- Analysez les données de performance des requêtes par programmation à l'aide de nouvelles API pour obtenir des statistiques d'exécution de requête, également publié aujourd'hui.

Pour savoir comment utiliser ces fonctionnalités dans le cadre de vos requêtes, regardez le didacticiel vidéo Optimize Amazon Athena Queries with New Query Analysis Tools sur la AWS YouTube chaîne.

Pour obtenir la documentation, consultez Affichage des plans d'exécution pour les requêtes SQL et Affichage des statistiques et des détails d'exécution pour les requêtes terminées.

11 juillet 2022

Date de publication : 11/07/2022

Vous pouvez désormais exécuter des requêtes paramétrées directement à partir de la console Athena ou de l'API sans préparer d'instructions SQL à l'avance.

Lorsque vous exécutez des requêtes dans la console Athena dont les paramètres se présentent sous la forme de points d'interrogation, l'interface utilisateur vous invite désormais à saisir directement des valeurs pour les paramètres. Cela évite de devoir modifier les valeurs littérales dans l'éditeur de requête chaque fois que vous souhaitez exécuter la requête.

Si vous utilisez l'API d'exécution de requêtes améliorée, vous pouvez désormais fournir les paramètres d'exécution et leurs valeurs en un seul appel.

Pour plus d'informations, consultez Utilisation de requêtes paramétrées dans ce guide de l'utilisateur et l'article du Big Data Blog intitulé AWS Utiliser des requêtes paramétrées Amazon Athena pour fournir des données en tant que service.

8 juillet 2022

Date de publication : 08/07/2022

Athena annonce les correctifs et améliorations suivants.

- Correction d'un problème lié à `DATE` la gestion de la conversion des colonnes pour les SageMaker points de terminaison (UDF) qui provoquait l'échec des requêtes.

6 juin 2022

Date de publication : 06/06/2022

Athena publie le pilote JDBC version 2.0.31. Le pilote JDBC 2.0.31 comprend les modifications suivantes :

- problème de dépendance log4j – Résolution d'un message d'erreur Cannot find driver class (Impossible de trouver une classe de pilote) causé par une dépendance log4j.

Pour plus d'informations et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, consultez Connexion à Amazon Athena avec JDBC.

25 mai 2022

Date de publication : 25/05/2022

Athena annonce les correctifs et améliorations suivants.

- Support Iceberg
  - Introduction d'un support pour les requêtes entre régions. Vous pouvez désormais interroger les tables Iceberg dans un Région AWS fichier différent de celui Région AWS que vous utilisez. Les requêtes entre régions ne sont pas prises en charge dans les régions chinoises.
  - Introduction d'un support pour la configuration du chiffrement côté serveur. Vous pouvez désormais utiliser SSE-S3/SSE-KMS pour chiffrer les données des opérations d'écriture Iceberg dans Amazon S3.

  Pour plus d'informations sur l'utilisation d'Apache Iceberg dans Athena, consultez Utilisation des tables Apache Iceberg.
- Publication du pilote JDBC 2.0.30

  Le pilote JDBC 2.0.30 pour Athena présente les améliorations suivantes :
  - Corrige un problème de course de données qui affectait les déclarations préparées paramétrisées.

- Corrige un problème de démarrage d'application qui survenait dans les environnements de création Gradle.

Pour télécharger le pilote JDBC 2.0.30, les notes de mise à jour et la documentation, consultez Connexion à Amazon Athena avec JDBC.

6 mai 2022

Date de publication : 06/05/2022

Publication des pilotes JDBC 2.0.29 et ODBC 1.1.17 pour Athena.

Ces pilotes comprennent les modifications suivantes :

- Mise à jour du processus de lancement du navigateur du plugin SAML.

Pour plus d'informations sur ces changements et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, veuillez consulter Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

22 avril 2022

Date de publication : 22/04/2022

Athena annonce les correctifs et améliorations suivants.

- Résolution d'un problème dans les index de partition et la fonction de filtrage avec le cache de partition qui s'est produit lorsque les conditions suivantes ont été remplies :
  - La `partition_filtering.enabled` clé a été définie sur `true` dans les AWS Glue propriétés d'une table.
  - La même table a été utilisée plusieurs fois avec des valeurs de filtre de partition différentes.

21 avril 2022

Date de publication : 21/04/2022

Vous pouvez désormais utiliser Amazon Athena pour exécuter des requêtes fédérées sur de nouvelles sources de données, notamment Google BigQuery, Azure Synapse et Snowflake. Les nouveaux connecteurs de source de données incluent :

- Stockage Azure Data Lake (ADLS) Gen2
- Azure Synapse
- Cloudera Hive
- Cloudera Impala
- Google BigQuery
- Hortonworks
- Microsoft SQL Server
- Oracle
- SAP HANA (Express Edition)
- Snowflake
- Teradata

Pour une liste complète des sources de données prises en charge par Athena, consultez Connecteurs de source de données disponibles.

Pour faciliter la navigation dans les sources disponibles et la connexion à vos données, vous pouvez désormais rechercher, trier et filtrer les connecteurs disponibles à partir d'une mise à jour des sources de données dans la console Athena.

Pour en savoir plus sur l'interrogation de sources fédérées, veuillez consulter Utilisation de la requête fédérée d'Amazon Athena et Exécution de requêtes fédérées.

13 avril 2022

Date de publication : 13/04/2022

Athena publie le pilote JDBC version 2.0.28. Le pilote JDBC 2.0.28 inclut les modifications suivantes :

- Support JWT – Le pilote prend désormais en charge les jetons web JSON (JWT) pour l'authentification. Pour plus d'informations sur l'utilisation de JWT avec le pilote JDBC, consultez le Guide d'installation et de configuration, téléchargeable depuis la page du pilote JDBC.
- Bibliothèque Log4j mise à jour – Le pilote JDBC utilise désormais les bibliothèques Log4j suivantes :
  - Log4j-api 2.17.1 (antérieurement 2.17.0)
  - Log4j-core 2.17.1 (antérieurement 2.17.0)
  - Log4j-jcl 2.17.2

- Autres améliorations – Le nouveau pilote inclut également les améliorations et corrections de bugs suivantes :
  - La fonctionnalité des déclarations préparées par Athena est désormais disponible via JDBC. Pour plus d'informations sur les instructions préparées, consultez Utilisation de requêtes paramétrées.
  - La fédération Athena JDBC SAML est désormais fonctionnelle pour les régions chinoises.
  - Améliorations mineures supplémentaires.

Pour plus d'informations et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, consultez Connexion à Amazon Athena avec JDBC.

30 mars 2022

Date de publication : 30/03/2022

Athena annonce les correctifs et améliorations suivants.

- Interrogation entre régions : vous pouvez désormais utiliser Athena pour interroger des données situées dans un compartiment Amazon S3, Régions AWS notamment en Asie-Pacifique (Hong Kong), au Moyen-Orient (Bahreïn), en Afrique (Le Cap) et en Europe (Milan). Les requêtes entre régions ne sont pas prises en charge dans les régions chinoises.
  - Pour obtenir la liste des sites Régions AWS dans lesquels Athena est disponible, consultez la section Points de terminaison et quotas Amazon Athena.
  - Pour plus d'informations sur l'activation et la désactivation par défaut, consultez la section Activation d'une région. Région AWS
  - Pour plus d'informations sur les requêtes entre régions, consultez Requête entre régions.

18 mars 2022

Date de publication : 18/03/2022

Athena annonce les correctifs et améliorations suivants.

- Dynamic filtering (Filtrage dynamique) – Dynamic filtering (Filtrage dynamique) a été amélioré pour les colonnes entières en appliquant efficacement le filtre à chaque registre d'une table correspondante.

- Iceberg — Correction d'un problème qui entraînait des échecs lors de l'écriture de fichiers Iceberg Parquet de plus de 2 Go.
- Uncompressed output (Sortie non compressée) – CREATE TABLE les instructions prennent désormais en charge l'écriture de fichiers non compressés. Pour écrire des fichiers non compressés, utilisez la syntaxe suivante :
  - CREATE TABLE (fichier texte ou JSON) — Dans `TBLPROPERTIES`, spécifiez `write.compression = NONE`.
  - CREATE TABLE (Parquet) — Dans `TBLPROPERTIES`, spécifiez `parquet.compression = UNCOMPRESSED`.
  - CREATE TABLE (ORC) — Dans `TBLPROPERTIES`, spécifiez `orc.compress = NONE`.
- Compression — Correction d'un problème lié aux insertions de tables de fichiers texte qui créaient des fichiers compressés dans un format mais qui utilisaient une autre extension de fichier de format de compression lorsque des méthodes de compression autres que par défaut étaient utilisées.
- Avro — Correction de problèmes survenus lors de la lecture de décimales de type fixe à partir de fichiers Avro.

2 mars 2022

Date de publication : 02/03/2022

Athena annonce les fonctions et améliorations suivantes.

- Vous pouvez désormais accorder au propriétaire du compartiment Simple Storage Service (Amazon S3) un accès total sur les résultats de la requête lorsque les listes de contrôle d'accès (ACL) sont activées pour le compartiment de résultats de la requête. Pour plus d'informations, consultez Spécification d'un emplacement de résultats de requête.
- Vous pouvez désormais mettre à jour les requêtes nommées existantes. Pour plus d'informations, consultez Utilisation de requêtes enregistrées.

23 février 2022

Date de publication : 23/02/2022

Athena annonce les correctifs et améliorations de performances suivants.

- Amélioration du traitement de la mémoire pour améliorer les performances et réduire les erreurs de mémoire.
- Athena lit désormais les colonnes d'horodatage ORC avec les informations de fuseau horaire stockées dans des pieds de page de bande et écrit des fichiers ORC avec fuseau horaire (UTC) dans les pieds de page. Cela n'affecte le comportement des lectures d'horodatage ORC que si le fichier ORC à lire a été créé dans un environnement de fuseau horaire non UTC.
- Correction des estimations incorrectes de la taille des tables de liens symboliques qui entraînaient des plans de requête sous-optimaux.
- Les vues éclatées latérales peuvent désormais être interrogées dans la console Athena à partir de sources de données de métastore Hive.
- Amélioration des messages d'erreur de lecture de Simple Storage Service (Amazon S3) pour inclure des informations plus détaillées sur les codes d'erreur de Simple Storage Service (Amazon S3).
- Correction d'un problème qui entraînait l'incompatibilité des fichiers de sortie au format ORC avec Apache Hive 3.1.
- Correction d'un problème qui entraînait l'échec des noms de table avec des guillemets dans certaines requêtes DML et DDL.

15 février 2022

Date de publication : 15/02/2022

Amazon Athena a augmenté le quota de requêtes DML actives dans toutes les régions. AWS Les requêtes actives incluent à la fois les requêtes en cours d'exécution et en file d'attente. Avec cette modification, vous pouvez désormais avoir plus de requêtes DML dans un état actif qu'auparavant.

Pour plus d'informations sur les quotas de service Athena, consultez Service Quotas. Pour connaître les quotas de requête dans la région où vous utilisez Athena, consultez Points de terminaison et quotas Amazon Athena dans la Références générales AWS.

Pour surveiller l'utilisation de vos quotas, vous pouvez utiliser les statistiques CloudWatch d'utilisation. Athena publie la métrique `ActiveQueryCount` dans l'espace de nom `AWS/Usage`. Pour plus d’informations, consultez Surveillance des métriques d'utilisation d'Athena.

Après avoir examiné votre utilisation, vous pouvez utiliser la console Service Quotas pour demander une augmentation de quota. Si vous avez précédemment demandé une augmentation de quota pour

votre compte, le quota demandé continue de s'appliquer s'il dépasse le nouveau quota de requête DML active par défaut. Sinon, tous les comptes utilisent la nouvelle valeur par défaut.

14 février 2022

Date de publication : 14/02/2022

Cette version ajoute le `ErrorType` sous-champ à l'objet de AthenaErrorréponse dans l'action d'API GetQueryExecutionAthena.

Alors que le champ `ErrorCategory` existant indique la source générale de l'échec d'une requête (système, utilisateur ou autre), le nouveau champ `ErrorType` fournit des informations plus précises sur l'erreur qui s'est produite. Combinez les informations des deux champs pour mieux comprendre les causes de l'échec de la requête.

Pour plus d’informations, consultez Catalogue d'erreurs Athena.

9 février 2022

Date de publication : 09/02/2022

L'ancienne console Athena n'est plus disponible. La nouvelle console d'Athena prend en charge toutes les fonctions de la console précédente, mais avec une interface plus facile à utiliser et moderne. Elle comprend de nouvelles fonctions qui améliorent l'expérience de développement de requêtes, d'analyse de données et de gestion de votre utilisation. Pour utiliser la nouvelle console Athena, rendez-vous sur https://console.aws.amazon.com/athena/.

8 février 2022

Date de publication : 08/02/2022

Propriétaire attendu du bucket : par mesure de sécurité supplémentaire, vous pouvez désormais éventuellement spécifier l' Compte AWS identifiant que vous pensez être le propriétaire du bucket d'emplacement de sortie des résultats de votre requête dans Athena. Si l'ID de compte du propriétaire du compartiment des résultats de la requête ne correspond pas à l'ID de compte que vous spécifiez, les tentatives de sortie vers le compartiment échoueront avec une erreur d'autorisation Simple Storage Service (Amazon S3). Vous pouvez définir ce paramètre au niveau du client ou du groupe de travail.

Pour plus d’informations, consultez Spécification d'un emplacement de résultats de requête.

28 janvier 2022

Date de publication : 28/01/2022

Athena annonce les améliorations suivantes des fonctions du moteur.

- Apache Hudi : les requêtes d'instantané sur les tables Hudi Merge on Read (MoR) peuvent désormais lire les colonnes d'horodatage qui ont le type de données `INT64`.
- Requêtes UNION : amélioration des performances et réduction de l'analyse des données pour certaines requêtes `UNION` qui analysent la même table plusieurs fois.
- Requêtes disjointes : amélioration des performances pour les requêtes qui ne comportent que des valeurs disjointes pour chaque colonne de partition du filtre.
- Améliorations de la projection de partition
  - Plusieurs valeurs disjointes sont désormais autorisées dans la condition de filtre pour les colonnes de type `injected`. Pour plus d'informations, consultez Type injecté.
  - Amélioration des performances pour les colonnes de types basés sur des chaînes comme `CHAR` ou `VARCHAR`, qui ne contiennent que des valeurs disjointes sur le filtre.

13 janvier 2022

Date de publication : 13/01/2022

Publication des pilotes JDBC 2.0.27 et ODBC 1.1.15 pour Athena.

Le pilote JDBC 2.0.27 inclut les modifications suivantes :

- Le pilote a été mis à jour pour récupérer des catalogues externes.
- Le numéro de version du pilote étendu est désormais inclus dans la chaîne `user-agent` dans le cadre de l'appel d'API Athena.

Le pilote ODBC 1.1.15 inclut les modifications suivantes :

- Corrige un problème lié aux seconds appels à `SQLParamData()`.

Pour plus d'informations sur ces changements et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, veuillez consulter Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

Notes de publication d'Athena pour 2021

26 novembre 2021

Date de publication : 26/11/2021

Athena annonce la version préliminaire publique des transactions Athena ACID, qui ajoutent des opérations d'écriture, de suppression, de mise à jour et de déplacement temporel au langage de manipulation des données (DML) SQL d'Athena. Les transactions Athena ACID permettent à plusieurs utilisateurs simultanés d'apporter des modifications fiables au niveau des lignes aux données Simple Storage Service (Amazon S3). Fondées sur le format de table Apache Iceberg, les transactions Athena ACID sont compatibles avec d'autres services et moteurs, tels que Amazon EMR et Apache Spark, qui prennent également en charge les formats de table Iceberg.

Les transactions Athena ACID et la syntaxe SQL familière simplifient les mises à jour de vos données commerciales et réglementaires. Par exemple, pour répondre à une demande d'effacement de données, vous pouvez effectuer une opération SQL `DELETE`. Pour effectuer des corrections d'enregistrement manuelles, vous pouvez utiliser une seule instruction `UPDATE`. Pour récupérer des données qui ont été récemment supprimées, vous pouvez émettre des requêtes Time Travel en utilisant une instruction `SELECT`. Les transactions Athena sont disponibles via la console d'Athena, les opérations API et les pilotes ODBC et JDBC.

Pour plus d'informations, consultez Utilisation des transactions Athena ACID.

24 novembre 2021

Date de publication : 24/11/2021

Athena annonce la prise en charge de la lecture et de l'écriture de données ORC, Parquet et de fichiers texte compressés selon la norme ZStandard. Athena utilise le niveau 3 de compression ZStandard lors de l'écriture de données compressées ZStandard.

Pour plus d'informations sur la compression des données dans Athena, veuillez consulter Prise en charge de la compression Athena.

22 novembre 2021

Date de publication : 22/11/2021

Vous pouvez désormais gérer les AWS Step Functions flux de travail depuis la console Amazon Athena, ce qui facilite la création de pipelines de traitement des données évolutifs, l'exécution de

requêtes basées sur une logique métier personnalisée, l'automatisation des tâches administratives et d'alerte, etc.

Step Functions est désormais intégré à la dernière génération de la console d'Athena, et vous pouvez l'utiliser pour visualiser un diagramme de flux interactif de vos machines à état qui invoquent Athena. Pour commencer, sélectionnez Workflows (Flux) dans le panneau de navigation de gauche. Si vous avez déjà des machines à états avec des requêtes Athena, sélectionnez une machine à états pour afficher un diagramme interactif du flux. Si vous débutez dans Step Functions, vous pouvez commencer en lançant un exemple de projet à partir de la console Athena et en le personnalisant en fonction de vos cas d'utilisation.

Pour plus d'informations, consultez Créer et orchestrer des pipelines ETL à l'aide d'Amazon Athena AWS Step Functions ou consultez la documentation Step Functions.

18 novembre 2021

Date de publication : 18/11/2021

Athena annonce de nouvelles fonctions et améliorations.

- Support spill-to-disk pour les requêtes d'agrégation contenant `DISTINCTORDER BY`, ou les deux, comme dans l'exemple suivant :

```
SELECT array_agg(orderstatus ORDER BY orderstatus)
FROM orders
GROUP BY orderpriority, custkey
```

- Résolution des problèmes de traitement de la mémoire pour les requêtes utilisant `DISTINCT`. Pour éviter les messages d'erreur tels que Query exhausted resources at this scale factor (La requête a épuisé les ressources à ce facteur d'échelle.)lorsque vous utilisez des requêtes `DISTINCT`, choisissez des colonnes dont la cardinalité est faible pour `DISTINCT`, ou réduisez la taille des données de la requête.
- Dans les requêtes `SELECT COUNT(*)` qui ne spécifient pas de colonne particulière, amélioration des performances et de l'utilisation de la mémoire en conservant uniquement le compte sans mise en mémoire tampon des lignes.
- Introduction des fonctions de chaîne suivantes.
  - `translate(source, from, to)` : renvoie la chaîne `source` avec les caractères présents dans la chaîne `from` remplacée par les caractères correspondants dans la chaîne `to`. Si la chaîne `from` contient des doublons, seule la première occurrence est utilisée. Si le caractère

`source` n'existe pas dans la chaîne `from`, le caractère `source` est copié sans traduction. Si l'index du caractère correspondant dans la chaîne `from` est supérieur à la longueur de la chaîne `to`, le caractère est omis de la chaîne résultante.

- `concat_ws(string0, array(varchar))` : renvoie la concaténation des éléments du tableau à l'aide de `string0` comme séparateur. Si `string0` a la valeur NULL, la valeur de retour est NULL. Toutes les valeurs NULL du tableau sont ignorées.

- Correction d'un bug dans lequel les requêtes échouaient lorsqu'elles tentaient d'accéder à un sous-champ manquant dans un `struct`. Les requêtes renvoient désormais une valeur NULL pour le sous-champ manquant.
- Correction d'un problème de hachage incohérent pour le type de données décimales.
- Correction d'un problème qui entraînait l'épuisement des ressources lorsqu'il y avait trop de colonnes dans une partition.

17 novembre 2021

Date de publication : 17/11/2021

Amazon Athena prend désormais en charge l'indexation des partitions pour accélérer les requêtes sur les tables partitionnées dans AWS Glue Data Catalog.

Lors de l'interrogation de tables partitionnées, Athena récupère et filtre les partitions de table disponibles vers le sous-ensemble correspondant à votre requête. À mesure que de nouvelles données et partitions sont ajoutées, il faut plus de temps pour traiter les partitions et le temps d'exécution des requêtes peut augmenter. Pour optimiser le traitement des partitions et améliorer les performances des requêtes sur des tables hautement partitionnées, Athena prend désormais en charge les index de partition AWS Glue.

Pour plus d'informations, consultez AWS Glue indexation et filtrage des partitions.

16 novembre 2021

Date de publication : 16/11/2021

La nouvelle console Amazon Athena améliorée est désormais généralement disponible dans les zones AWS commerciales et dans les GovCloud régions où Athena est disponible. La nouvelle console d'Athena prend en charge toutes les fonctions de la console précédente, mais avec une interface plus facile à utiliser et moderne. Elle comprend de nouvelles fonctions qui améliorent

l'expérience de développement de requêtes, d'analyse de données et de gestion de votre utilisation. Vous pouvez désormais :

- Réorganiser, accéder à ou fermer plusieurs onglets de requête à partir d'une barre d'onglets de requête redessinée.
- Lire et modifier les requêtes plus facilement grâce à une mise en forme améliorée du code SQL et du texte.
- Copier les résultats de la requête dans votre presse-papiers en plus de télécharger le jeu de résultats complet.
- Trier l'historique de vos requêtes, vos requêtes enregistrées et vos groupes de travail, et choisir les colonnes à afficher ou à masquer.
- Utiliser une interface simplifiée pour configurer les sources de données et les groupes de travail en moins de clics.
- Définir les préférences d'affichage des résultats de la requête, de l'historique des requêtes, de l'encapsulation des lignes, etc.
- Augmenter votre productivité grâce à des nouveaux et meilleurs raccourcis clavier et à la documentation produit intégrée.

Avec l'annonce d'aujourd'hui, la console repensée est désormais celle par défaut. Pour nous parler de votre expérience, choisissez Feedback (Commentaire) dans le coin inférieur gauche de la console.

Si vous le souhaitez, vous pouvez utiliser la console précédente en vous connectant à votre console Compte AWS, en choisissant Amazon Athena et en désélectionnant New Athena Experience dans le panneau de navigation de gauche.

12 novembre 2021

Date de publication : 12/11/2021

Vous pouvez désormais utiliser Amazon Athena pour exécuter des requêtes fédérées sur des sources de données situées dans un autre compte AWS que le vôtre. Jusqu'à aujourd'hui, l'interrogation de ces données nécessitait que la source de données et son connecteur utilisent les mêmes informations Compte AWS que l'utilisateur qui a demandé les données.

En tant qu'administrateur de données, vous pouvez activer les requêtes fédérées entre comptes en partageant votre connecteur de données avec le compte d'un analyste de données. En tant

qu'analyste de données, vous pouvez ajouter un connecteur de données qu'un administrateur de données a partagé avec vous à votre compte. Les modifications de configuration apportées au connecteur dans le compte d'origine s'appliquent automatiquement au connecteur partagé.

Pour plus d'informations sur l'activation des requêtes fédérées entre comptes, veuillez consulter Activation des requêtes fédérées entre comptes. Pour en savoir plus sur l'interrogation de sources fédérées, veuillez consulter Utilisation de la requête fédérée d'Amazon Athena et Exécution de requêtes fédérées.

2 novembre 2021

Date de publication : 02/11/2021

Vous pouvez désormais utiliser l'instruction `EXPLAIN ANALYZE` dans Athena pour visualiser le plan d'exécution distribué et le coût de chaque opération pour vos requêtes SQL.

Pour plus d'informations, consultez Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena.

29 octobre 2021

Date de publication : 29/10/2021

Athena publie les pilotes JDBC 2.0.25 et ODBC 1.1.13 et annonce des fonctions et des améliorations.

Pilotes JDBC et ODBC

Publication des pilotes JDBC 2.0.25 et ODBC 1.1.13 pour Athena. Les deux pilotes prennent en charge l'authentification multifacteur SAML du navigateur, qui peut être configurée pour fonctionner avec n'importe quel fournisseur SAML 2.0.

Le pilote JDBC 2.0.25 inclut les modifications suivantes :

- Support de l'authentification SAML du navigateur. Le pilote inclut un plugin SAML de navigateur qui peut être configuré pour fonctionner avec n'importe quel fournisseur SAML 2.0.
- Support pour les appels AWS Glue d'API. Vous pouvez utiliser le paramètre `GlueEndpointOverride` pour remplacer le point de terminaison AWS Glue .
- Modification du classpath de `com.simba.athena.amazonaws` à `com.amazonaws`.

Le pilote ODBC 1.1.13 inclut les modifications suivantes :

- Support de l'authentification SAML du navigateur. Le pilote inclut un plugin SAML de navigateur qui peut être configuré pour fonctionner avec n'importe quel fournisseur SAML 2.0. Pour obtenir un exemple d'utilisation du plugin SAML du navigateur avec le pilote ODBC, veuillez consulter Configuration de l'authentification unique à l'aide d'ODBC, SAML 2.0 et du fournisseur d'identité Okta.
- Vous pouvez désormais configurer la durée de la session de rôle lorsque vous utilisez ADFS, Azure AD ou Navigateur Azure AD pour l'authentification.

Pour plus d'informations sur ces changements et d'autres, et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, veuillez consulter Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

Fonctionnalités et améliorations

Athena annonce les fonctions et améliorations suivantes.

- Une nouvelle règle d'optimisation a été introduite pour éviter les analyses de tables en double dans certains cas.

4 octobre 2021

Date de publication : 04/10/2021

Athena annonce les fonctions et améliorations suivantes.

- DÉCALAGE SQL : la clause SQL `OFFSET` est désormais prise en charge dans les instructions `SELECT`. Pour plus d'informations, consultez SELECT.
- CloudWatch métriques d'utilisation — Athena publie désormais la `ActiveQueryCount` métrique dans l'espace de `AWS/Usage` noms. Pour plus d'informations, consultez Surveillance des métriques d'utilisation d'Athena.
- Planification des requêtes : correction d'un bug qui pouvait, dans de rares cas, entraîner des délais d'expiration de la planification des requêtes.

16 septembre 2021

Date de publication : 16/09/2021

Athena annonce les nouvelles fonctions et améliorations suivantes.

Fonctionnalités

- Ajout de la prise en charge de la spécification du fichier texte et de la compression JSON dans CTAS à l'aide de la propriété de table `write_compression`. Vous pouvez également spécifier la propriété `write_compression` dans CTAS pour les formats Parquet et ORC. Pour plus d'informations, consultez Propriétés de la table CTAS.
- Le format de compression BZIP2 est désormais pris en charge pour l'écriture de fichiers texte et de fichiers JSON. Pour plus d'informations sur les formats de compression dans Athena, veuillez consulter Prise en charge de la compression Athena.

Améliorations

- Correction d'un bug dans lequel les informations d'identité ne pouvaient pas être envoyées à la fonction Lambda UDF.
- Correction d'un problème de poussée des prédicats avec des conditions de filtre disjointes.
- Correction d'un problème de hachage pour les types décimaux.
- Correction d'un problème de collecte inutile de statistiques.
- Suppression d'un message d'erreur incohérent.
- Amélioration des performances de la jointure par diffusion en appliquant un élagage dynamique des partitions dans le composant master.
- Pour les requêtes fédérées :
  - Modification de la configuration pour réduire l'occurrence des erreurs `CONSTRAINT_VIOLATION` dans les requêtes fédérées.

15 septembre 2021

Date de publication : 15/09/2021

Vous pouvez désormais utiliser une console Amazon Athena repensée (version préliminaire). Un nouveau pilote Athena JDBC a été publié.

Version préliminaire de la console Athena

Vous pouvez désormais utiliser une console Amazon Athena redessinée (version préliminaire) depuis n'importe quel Région AWS endroit où Athena est disponible. La nouvelle console prend en charge toutes les fonctions de la console existante, mais depuis une interface moderne et plus facile à utiliser.

Pour passer à la nouvelle console, connectez-vous à votre console Compte AWS et choisissez Amazon Athena. Dans la barre de navigation de la AWS console, choisissez Basculer vers la nouvelle console. Pour revenir à la console par défaut, désélectionnez New Athena experience (Nouvelle expérience Athena) à partir du panneau de navigation de gauche.

Commencez dès aujourd'hui avec la nouvelle console. Choisissez Feedback (Commentaire) dans le coin inférieur gauche pour nous parler de votre expérience.

Pilote Athena JDBC 2.0.24

Athena annonce la disponibilité du pilote JDBC version 2.0.24 pour Athena. Cette version met à jour la prise en charge du proxy pour tous les fournisseurs d'informations Le pilote prend désormais en charge l'authentification par proxy pour tous les hôtes qui ne sont pas pris en charge par la propriété de connexion `NonProxyHosts`.

Pour des raisons pratiques, cette version inclut le téléchargement du pilote JDBC avec et sans le AWS SDK. Cette version du pilote JDBC vous permet d'avoir à la fois le kit SDK AWS et le pilote JDBC Athena intégrés dans le projet.

Pour plus d'informations et pour télécharger le nouveau pilote, les notes de mise à jour et la documentation, voir Connexion à Amazon Athena avec JDBC.

31 août 2021

Date de publication : 31/08/2021

Athena annonce les améliorations de fonctions et les corrections de bogues suivantes.

- Améliorations de la fédération Athena : athena a ajouté la prise en charge des types de cartes et une meilleure prise en charge des types complexes dans le cadre du kit Athena Query Federation SDK. Cette version comprend également des améliorations de la mémoire et des optimisations des performances.

- Nouvelles catégories d'erreurs : introduction des catégories d'erreur `USER` et `SYSTEM` dans les messages d'erreur. Ces catégories vous aident à distinguer les erreurs que vous pouvez corriger vous-même (`USER`) et les erreurs qui peuvent nécessiter l'assistance du support Athena (`SYSTEM`).
- Messagerie d'erreur de requête fédérée : mise à jour des catégorisations `USER_ERROR` pour les erreurs liées aux requêtes fédérées.
- JOIN — Correction de bogues et de problèmes de mémoire spill-to-disk connexes afin d'améliorer les performances et de réduire les erreurs de mémoire lors `JOIN` des opérations.

12 août 2021

Date de publication : 12/08/2021

Publication du pilote ODBC 1.1.12 pour Athena. Cette version corrige les problèmes liés à `SQLPrepare()`, `SQLGetInfo()` et `EndpointOverride`.

Pour télécharger le nouveau pilote, les notes de mise à jour et la documentation, voir Connexion à Amazon Athena avec le pilote ODBC.

6 août 2021

Date de publication : 06/08/2021

Amazon Athena annonce la disponibilité d'Athena et ses fonctions dans la région Asie-Pacifique (Osaka).

Cette version étend la disponibilité d'Athena dans la région Asie-Pacifique pour inclure Asie-Pacifique (Hong Kong), Asie-Pacifique (Mumbai), Asie-Pacifique (Osaka), Asie-Pacifique (Séoul), Asie-Pacifique (Singapour), Asie-Pacifique (Sydney) et Asie-Pacifique (Sydney) et Asie-Pacifique (Tokyo). Pour une liste complète des services Services AWS disponibles dans ces régions et dans d'autres, consultez la liste Région AWS complète des services.

5 août 2021

Date de publication : 05/08/2021

Vous pouvez utiliser l'instruction `UNLOAD` pour écrire la sortie d'une requête `SELECT` dans les formats PARQUET, ORC, AVRO et JSON.

Pour plus d’informations, consultez UNLOAD.

30 juillet 2021

Date de publication : 30/07/2021

Athena annonce les améliorations de fonctions et les corrections de bogues suivantes.

- Filtrage dynamique et élagage des partitions : ces améliorations permettent d'augmenter les performances et de réduire la quantité de données analysées dans certaines requêtes, comme dans l'exemple suivant.

  Cet exemple suppose que `Table_B` est une table non partitionnée dont la taille des fichiers est inférieure à 20 Mo. Pour les requêtes de ce type, moins de données sont lues à partir de la `Table_A` et la requête se termine plus rapidement.

  ```
  SELECT *
  FROM Table_A
  JOIN Table_B ON Table_A.date = Table_B.date
  WHERE Table_B.column_A = "value"
  ```

- ORDER BY avec LIMIT, DISTINCT with LIMIT : amélioration des performances des requêtes utilisant `ORDER BY` ou `DISTINCT` suivies d'une clause `LIMIT`.
- Fichiers S3 Glacier Deep Archive : lorsqu'Athena interroge une table contenant à la fois des fichiers S3 Glacier Deep Archive et des fichiers non S3 Glacier, Athena ignore désormais les fichiers S3 Glacier Deep Archive. Auparavant, vous deviez déplacer manuellement ces fichiers depuis l'emplacement de la requête, faute de quoi la requête échouait. Si vous souhaitez utiliser Athena pour interroger des objets dans la mémoire me stockage S3 Glacier Deep Archive, vous devez les restaurer. Pour plus d'informations, consultez la rubrique Restauration d'un objet archivé du Guide de l'utilisateur de Simple Storage Service (Amazon S3).
- Correction d'un bogue qui faisait que les fichiers vides créés par la propriété de table `bucketed_by` CTAS n'étaient pas chiffrés correctement.

21 juillet 2021

Date de publication : 21/07/2021

Avec la version de juillet 2021 de Microsoft Power BI Desktop, vous pouvez créer des rapports et des tableaux de bord en utilisant un connecteur de source de données natif pour Amazon Athena. Le connecteur pour Amazon Athena est disponible en tant que connecteur standard dans Power BI. Il

prend en charge DirectQueryet permet l'analyse de grands ensembles de données et l'actualisation du contenu via Power BI Gateway.

Étant donné que le connecteur utilise votre nom de source de données (DSN) ODBC existant pour se connecter à Athena et exécuter des requêtes sur Athena, il nécessite le pilote ODBC Athena. Pour télécharger le dernier pilote ODBC, voir Connexion à Amazon Athena avec le pilote ODBC.

Pour plus d'informations, consultez Utilisation du connecteur Amazon Athena Power BI.

16 juillet 2021

Date de publication : 16/07/2021

Amazon Athena a mis à jour son intégration à Apache Hudi. Hudi est un cadre de gestion de données open source utilisé pour simplifier le traitement progressif des données dans les lacs de données Simple Storage Service (Amazon S3). L'intégration mise à jour vous permet d'utiliser Athena pour interroger les tables Hudi 0.8.0 gérées par Amazon EMR, Apache Spark, Apache Hive ou d'autres services compatibles. En outre, Athena prend désormais en charge deux nouvelles fonctions : les requêtes d'instantané sur les tables de type « fusion sur lecture » (MoR, Merge-on-Read) et la prise en charge de la lecture sur les tables amorcées.

Apache Hudi permet le traitement des données au niveau des registres, ce qui peut vous aider à simplifier le développement des pipelines de capture de données modifiées (CDC, Change Data Capture), à vous conformer aux mises à jour et aux suppressions imposées par le RGPD et à mieux gérer les données diffusées en streaming provenant de capteurs ou de dispositifs qui nécessitent l'insertion de données et la mise à jour d'événements. La version 0.8.0 facilite la migration des grandes tables Parquet vers Hudi sans copier les données afin de pouvoir les interroger et les analyser via Athena. Vous pouvez utiliser la nouvelle prise en charge des requêtes d'instantané d'Athena pour obtenir des vues en temps quasi réel des mises à jour de vos tables diffusées en streaming.

Pour en savoir plus sur l'utilisation de Hudi avec Athena, voir Utilisation d'Athena pour interroger des jeux de données Apache Hudi.

8 juillet 2021

Date de publication : 08/07/2021

Publication du pilote ODBC 1.1.11 pour Athena. Le pilote ODBC peut désormais authentifier la connexion à l'aide d'un jeton Web JSON (JWT). Sous Linux, la valeur par défaut de la propriété Groupe de travail a été définie sur Primaire.

Pour plus d'informations et pour télécharger le nouveau pilote, les notes de mise à jour et la documentation, voir Connexion à Amazon Athena avec le pilote ODBC.

1er juillet 2021

Date de publication : 01/07/2021

Le 1er juillet 2021, le traitement spécial des groupes de travail de prévisualisation a pris fin. Bien que les groupes de travail `AmazonAthenaPreviewFunctionality` retiennent leur nom, ils n'ont plus de statut spécial. Vous pouvez continuer à utiliser les groupes de travail `AmazonAthenaPreviewFunctionality` pour visualiser, modifier, organiser et exécuter des requêtes. Toutefois, les requêtes qui utilisent des fonctions qui étaient auparavant en prévisualisation sont désormais soumises aux conditions de facturation standard d'Athena. Pour plus d'informations sur la facturation, consultez la rubrique Tarification Amazon Athena.

23 Juin 2021

Date de publication : 23/06/2021

Publication des pilotes JDBC 2.0.23 et ODBC 1.1.10 pour Athena. Les deux pilotes offrent des performances de lecture améliorées et prennent en charge les instructions EXPLAIN et les requêtes paramétrées.

Les instructions `EXPLAIN` montrent le plan d'exécution logique ou distribué d'une requête SQL. Les requêtes paramétrées permettent d'utiliser la même requête plusieurs fois avec des valeurs différentes fournies au moment de l'exécution.

La version JDBC ajoute également la prise en charge d'Active Directory Federation Services 2019 et une option de remplacement du point de terminaison personnalisé pour AWS STS. La version ODBC corrige un problème avec les informations d'identification du profil IAM.

Pour plus d'informations et pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, voir Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

12 mai 2021

Date de publication : 12/05/2021

Vous pouvez désormais utiliser Amazon Athena pour enregistrer un AWS Glue catalogue à partir d'un compte autre que le vôtre. Après avoir configuré les autorisations IAM requises pour AWS Glue, vous pouvez utiliser Athena pour exécuter des requêtes entre comptes.

Pour plus d'informations, consultez Enregistrer un compte AWS Glue Data Catalog depuis un autre compte et Accès entre comptes aux catalogues de données AWS Glue.

10 mai 2021

Date de publication : 10/05/2021

Publication de la version 1.1.9.1001 du pilote ODBC pour Athena. Cette version corrige un problème avec le type d'authentification `BrowserAzureAD` lors de l'utilisation d'Azure Active Directory (AD).

Pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, voir Connexion à Amazon Athena avec le pilote ODBC.

5 mai 2021

Date de publication : 05/05/2021

Vous pouvez désormais utiliser le connecteur Vertica d'Amazon Athena dans les requêtes fédérées pour interroger les sources de données Vertica depuis Athena. Par exemple, vous pouvez exécuter des requêtes analytiques sur un entrepôt de données sur Vertica et un lac de données sur Simple Storage Service (Amazon S3).

Pour déployer le connecteur Athena Vertica, rendez-vous AthenaVerticaConnectorsur la page du. AWS Serverless Application Repository

Le connecteur Vertica d'Amazon Athena expose plusieurs options de configuration par le biais de variables d'environnement Lambda. Pour plus d'informations sur les options de configuration, les paramètres, les chaînes de connexion, le déploiement et les limitations, voir Connecteur Amazon Athena pour Vertica.

Pour obtenir des informations détaillées sur l'utilisation du connecteur Vertica, consultez la rubrique Interrogation d'une source de données Vertica dans Amazon Athena à l'aide du kit SDK de requête fédérée d'Athena sur le blog AWS Big Data.

30 avril 2021

Date de publication : 30/04/2021

Publication des pilotes JDBC 2.0.21 et ODBC 1.1.9 pour Athena. Les deux versions prennent en charge l'authentification SAML avec Azure Active Directory (AD) et l'authentification SAML avec. PingFederate La version JDBC prend également en charge les requêtes paramétrées. Pour plus d'informations sur les requêtes paramétrées dans Athena, voir Utilisation de requêtes paramétrées.

Pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, voir Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

29 avril 2021

Date de publication : 29/04/2021

Amazon Athena annonce la disponibilité de la version 2 du moteur Athena dans les régions Chine (Beijing) et Chine (Ningxia).

Pour plus d'informations sur la version 2 du moteur Athena, voir Version 2 du moteur Athena.

26 avril 2021

Date de publication : 26/04/2021

Les fonctions de valeur de fenêtre dans la version 2 du moteur Athena prennent désormais en charge `IGNORE NULLS` et `RESPECT NULLS`.

Pour plus d'informations, consultez la rubrique Fonctions de valeur dans la documentation Presto.

21 avril 2021

Date de publication : 21/04/2021

Amazon Athena annonce la disponibilité de la version 2 du moteur Athena dans les régions Europe (Milan) et Afrique (Le Cap).

Pour plus d'informations sur la version 2 du moteur Athena, voir Version 2 du moteur Athena.

5 avril 2021

Date de publication : 05/04/2021

Instruction EXPLAIN

Vous pouvez maintenant utiliser l'instruction `EXPLAIN` dans Athena pour visualiser le plan d'exécution de vos requêtes SQL.

Pour plus d'informations, consultez Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena et Explication des résultats de l'instruction EXPLAIN d'Athena.

SageMaker Modèles de Machine Learning dans les requêtes SQL

L'inférence de modèles d'apprentissage automatique avec Amazon SageMaker est désormais généralement disponible pour Amazon Athena. Utilisez des modèles de machine learning dans des requêtes SQL pour simplifier des tâches complexes telles que la détection d'anomalies, l'analyse de cohortes de clients et les prédictions de séries temporelles en invoquant une fonction dans une requête SQL.

Pour plus d'informations, consultez Utilisation du Machine Learning (ML) avec Amazon Athena.

Fonctions définies par l'utilisateur (UDF)

Les fonctions définies par l'utilisateur (UDF) sont désormais généralement disponibles pour Athena. Utilisez les UDF pour exploiter des fonctions personnalisées qui traitent des registres ou des groupes de registres dans une seule requête SQL.

Pour plus d'informations, consultez Interrogation avec des fonctions définies par l'utilisateur.

30 mars 2021

Date de publication : 30/03/2021

Amazon Athena annonce la disponibilité de la version 2 du moteur Athena dans les régions Asie-Pacifique (Hong Kong) et Moyen-Orient (Bahreïn).

Pour plus d'informations sur la version 2 du moteur Athena, voir Version 2 du moteur Athena.

25 mars 2021

Date de publication : 25/03/2021

Amazon Athena annonce la disponibilité de la version 2 du moteur Athena dans la région Europe (Stockholm).

Pour plus d'informations sur la version 2 du moteur Athena, voir Version 2 du moteur Athena.

5 mars 2021

Date de publication : 05/03/2021

Amazon Athena annonce la disponibilité de la version 2 du moteur Athena dans les régions Canada (Centre), Europe (Francfort) et Amérique du Sud (Sao Paulo).

Pour plus d'informations sur la version 2 du moteur Athena, voir Version 2 du moteur Athena.

25 février 2021

Date de publication : 25/02/2021

Amazon Athena annonce la disponibilité générale de la version 2 du moteur Athena dans les régions Asie-Pacifique (Séoul), Asie-Pacifique (Singapour), Asie-Pacifique (Sydney), Europe (Londres) et Europe (Paris).

Pour plus d'informations sur la version 2 du moteur Athena, voir Version 2 du moteur Athena.

Notes de publication d'Athena pour 2020

16 décembre 2020

Date de publication : 16/12/2020

Amazon Athena annonce la disponibilité de la version 2 du moteur Athena, Athena Federated Query, et dans d'autres régions. AWS PrivateLink

Version 2 du moteur Athena et requête fédérée d’Athena

Amazon Athena annonce la disponibilité générale de la version 2 du moteur Athena et de la requête fédérée d’Athena dans les régions Asie-Pacifique (Mumbai), Asie-Pacifique (Tokyo), Europe (Irlande) et USA Ouest (Californie du Nord). La version 2 du moteur Athena et les requêtes fédérées sont déjà disponibles dans les régions USA Est (Virginie du Nord), USA Est (Ohio) et USA Ouest (Oregon).

Pour plus d’informations, consultez Version 2 du moteur Athena et Utilisation de la requête fédérée d’Amazon Athena.

AWS PrivateLink

AWS PrivateLink for Athena est désormais pris en charge dans la région Europe (Stockholm). Pour plus d'informations sur AWS PrivateLink Athéna, voir. Connexion à Amazon Athena à l'aide d'un point de terminaison de VPC d'interface

24 novembre 2020

Date de publication : 24/11/2020

Publication des pilotes JDBC 2.0.16 et ODBC 1.1.6 pour Athena. Ces versions prennent en charge, au niveau du compte, l'authentification multifactorielle (MFA) Okta Verify. Vous pouvez également utiliser Okta MFA pour configurer l'authentification SMS et l'authentification Google Authenticator en tant que facteurs.

Pour télécharger les nouveaux pilotes, les notes de mise à jour et la documentation, voir Connexion à Amazon Athena avec JDBC et Connexion à Amazon Athena avec le pilote ODBC.

11 novembre 2020

Date de publication : 11/11/2020

Amazon Athena annonce la disponibilité générale de la version 2 du moteur Athena et des requêtes fédérées dans les régions USA Est (Virginie du Nord), USA Est (Ohio) et USA Ouest (Oregon).

Version 2 du moteur Athena

Amazon Athena annonce la disponibilité générale d'une nouvelle version du moteur de requête, la version 2 du moteur Athena, dans les régions USA Est (Virginie du Nord), USA Est (Ohio) et USA Ouest (Oregon).

La version 2 du moteur Athena comprend des améliorations des performances et de nouvelles fonctions telles que la prise en charge de l'évolution des schémas pour les données au format Parquet, des fonctions géospatiales supplémentaires, la prise en charge de la lecture de schémas imbriqués pour réduire les coûts et des améliorations des performances des opérations JOIN et AGGREGATE.

- Pour plus d'informations sur les améliorations, les évolutions et les corrections de bogues, voir Version 2 du moteur Athena.
- Pour plus d'informations sur la procédure de mise à niveau, voir Modification des versions du moteur Athena.

- Pour plus d'informations sur le test des requêtes, voir Essai des requêtes avant la mise à jour d'une version du moteur.

Requêtes SQL fédérées

Vous pouvez désormais utiliser la requête fédérée d'Athena dans les régions USA Est (Virginie du Nord), USA Est (Ohio) et USA Ouest (Oregon) sans utiliser le groupe de travail `AmazonAthenaPreviewFunctionality`.

Utilisez les requêtes SQL fédérées pour exécuter des requêtes SQL sur des sources de données relationnelles, non relationnelles, objet et personnalisées. Grâce aux requêtes fédérées, vous pouvez soumettre une seule requête SQL qui analyse les données provenant de plusieurs sources exécutées sur site ou hébergées dans le cloud.

L'exécution d'analyses sur les données réparties entre les applications peut être complexe et chronophage pour les raisons suivantes :

- Les données nécessaires aux analyses sont souvent réparties dans des magasins de données relationnels, valeurs clés, de documents, en mémoire, de recherche, de graphiques, d'objets, de séries chronologiques et de grand livre.
- Pour analyser les données provenant de ces sources, les analystes construisent des pipelines complexes pour extraire, transformer et charger un entrepôt de données afin que les données puissent être interrogées.
- L'accès aux données provenant de différentes sources nécessite l'apprentissage de nouveaux langages de programmation et de nouveaux concepts d'accès aux données.

Les requêtes SQL fédérées dans Athena éliminent cette complexité en permettant aux utilisateurs d'interroger les données sur place, où qu'elles se trouvent. Les analystes peuvent utiliser des structures SQL familières pour joindre (`JOIN`) des données à plusieurs sources de données pour une analyse rapide et stocker les résultats dans Simple Storage Service (Amazon S3) pour une utilisation ultérieure.

Connecteurs de source de données

Pour traiter les requêtes fédérées, Athena utilise les connecteurs de sources de données Athena qui s'exécutent sur AWS Lambda. Les connecteurs open source et prédéfinis suivants ont été écrits et testés par Athena. Utilisez-les pour exécuter des requêtes SQL dans Athena sur leurs sources de données correspondantes.

- CloudWatch
- Métriques CloudWatch
- DocumentDB
- DynamoDB
- OpenSearch
- HBase
- Neptune
- Redis
- Timestream
- TPC Benchmark DS (TPC-DS)

Connecteurs de source de données personnalisés

Grâce au kit Athena Query Federation SDK, les développeurs peuvent créer des connecteurs à n'importe quelle source de données pour permettre à Athena d'exécuter des requêtes SQL sur cette source de données. Athena Query Federation Connector étend les avantages des requêtes fédérées au-delà des connecteurs fournis. AWS Comme les connecteurs fonctionnent en AWS Lambda continu, vous n'avez pas à gérer l'infrastructure ni à planifier l'adaptation aux pics de demande.

Étapes suivantes

- Pour en savoir plus sur la fonction de requête fédérée, voir Utilisation de la requête fédérée d'Amazon Athena.
- Pour commencer à utiliser un connecteur existant, reportez-vous à la section Déploiement d'un connecteur et connexion à une source de données.
- Pour savoir comment créer votre propre connecteur de source de données à l'aide du SDK Athena Query Federation, consultez Example Athena Connector on. GitHub

22 octobre 2020

Date de publication : 22/10/2020

Tu peux maintenant appeler Athéna avec. AWS Step Functions AWS Step Functions peut contrôler certains Services AWS directement à l'aide de l'Amazon States Language. Vous pouvez utiliser

Step Functions avec Athena pour lancer et arrêter l'exécution de requêtes, obtenir des résultats de requêtes, exécuter des requêtes de données ad hoc ou planifiées et récupérer les résultats des lacs de données dans Amazon S3.

Pour plus d'informations, consultez la rubrique Appel d'Athena avec Step Functions du Guide du développeur AWS Step Functions .

29 juillet 2020

Date de publication : 29/07/2020

Publication du pilote JDBC version 2.0.13. Cette version prend en charge l'utilisation de plusieurs catalogues de données enregistrés dans Athena, le service Okta pour l'authentification et les connexions aux points de terminaison de VPC.

Pour télécharger et utiliser la nouvelle version du pilote, voir Connexion à Amazon Athena avec JDBC.

9 juillet 2020

Date de publication : 09/07/2020

Amazon Athena prend en charge l'interrogation des ensembles de données Hudi compactés et ajoute la AWS CloudFormation `AWS::Athena::DataCatalog` ressource permettant de créer, de mettre à jour ou de supprimer les catalogues de données que vous enregistrez dans Athena.

Jeux de données Apache Hudi

Apache Hudi est un cadre de gestion de données open source qui simplifie le traitement progressif des données. Amazon Athena prend désormais en charge l'interrogation de la vue optimisée en lecture d'un jeu de données Apache Hudi dans votre lac de données basé sur Simple Storage Service (Amazon S3).

Pour plus d’informations, consultez Utilisation d'Athena pour interroger des jeux de données Apache Hudi.

AWS CloudFormation Ressource de catalogue de données

Pour utiliser la fonction de requête fédérée d'Amazon Athena afin d'interroger n'importe quelle source de données, vous devez d'abord enregistrer votre catalogue de données dans Athena. Vous pouvez

désormais utiliser cette AWS CloudFormation `AWS::Athena::DataCatalog` ressource pour créer, mettre à jour ou supprimer les catalogues de données que vous enregistrez dans Athena.

Pour plus d'informations, consultez AWS::Athena::DataCatalogle guide de AWS CloudFormation l'utilisateur.

1er juin 2020

Date de publication : 01/06/2020

Utilisation du métastore Apache Hive comme métacatalogue avec Amazon Athena

Vous pouvez désormais connecter Athena à un ou plusieurs métastores Apache Hive en plus du AWS Glue Data Catalog avec Athena.

Pour vous connecter à un métastore Hive auto-hébergé, vous avez besoin d'un connecteur de métastore Hive Athena. Athena fournit un connecteur de mise en œuvre de référence que vous pouvez utiliser. Le connecteur s'exécute en tant que fonction AWS Lambda dans votre compte.

Pour plus d'informations, consultez Utilisation du connecteur de données Athena pour un métastore Hive externe.

21 mai 2020

Date de publication : 21/05/2020

Amazon Athena ajoute la prise en charge de la projection de partition. Utilisez la projection de partition pour accélérer le traitement des requêtes de tables hautement partitionnées et automatiser la gestion des partitions. Pour plus d'informations, consultez Projection de partition avec Amazon Athena.

1er avril 2020

Date de publication : 01/04/2020

Outre la Région USA Est (Virginie du Nord), les fonctions de requête fédérée d'Amazon Athena, les fonctions définies par l'utilisateur (UDF), l'inférence machine learning et le métastore Hive externe sont désormais disponibles en version préliminaire dans les Régions Asie-Pacifique (Mumbai), Europe (Irlande) et USA Ouest (Oregon).

11 mars 2020

Date de publication : 11/03/2020

Amazon Athena publie désormais des EventBridge événements Amazon pour les transitions d'état des requêtes. Lorsqu'une requête passe d'un état à un autre (par exemple, de l'état En cours à un état terminal tel que Réussi ou Annulé), Athena publie un événement de changement d'état de requête sur. EventBridge Cet événement contient des informations sur le changement de l'état de la requête. Pour plus d'informations, consultez Surveillance des requêtes Athena à l'aide des événements Amazon EventBridge.

6 mars 2020

Date de publication : 06/03/2020

Vous pouvez désormais créer et mettre à jour des groupes de travail Amazon Athena à l'aide de cette ressource. AWS CloudFormation `AWS::Athena::WorkGroup` Pour plus d'informations, consultez AWS::Athena::WorkGrouple guide de AWS CloudFormation l'utilisateur.

Notes de publication d'Athena pour 2019

26 novembre 2019

Date de publication : 17/12/2019

Amazon Athena s'enrichit de la prise en charge de l'exécution de requêtes SQL sur des sources de données relationnelles, non relationnelles, objet et personnalisées, l'invocation de modèles de machine learning dans des requêtes SQL, des fonctions définies par l'utilisateur (UDF) (Prévisualisation), l'utilisation du métastore Apache Hive comme catalogue de métadonnées avec Amazon Athena (Prévisualisation), ainsi que quatre métriques supplémentaires liées aux requêtes.

Requêtes SQL fédérées

Utilisez les requêtes SQL fédérées pour exécuter des requêtes SQL sur des sources de données relationnelles, non relationnelles, objet et personnalisées.

Vous pouvez désormais utiliser la requête fédérée d'Athena pour analyser les données stockées dans des sources de données relationnelles, non relationnelles, objet et personnalisées. Grâce aux requêtes fédérées, vous pouvez soumettre une seule requête SQL qui analyse les données provenant de plusieurs sources exécutées sur site ou hébergées dans le cloud.

L'exécution d'analyses sur les données réparties entre les applications peut être complexe et chronophage pour les raisons suivantes :

- Les données nécessaires aux analyses sont souvent réparties dans des magasins de données relationnels, valeurs clés, de documents, en mémoire, de recherche, de graphiques, d'objets, de séries chronologiques et de grand livre.
- Pour analyser les données provenant de ces sources, les analystes construisent des pipelines complexes pour extraire, transformer et charger un entrepôt de données afin que les données puissent être interrogées.
- L'accès aux données provenant de différentes sources nécessite l'apprentissage de nouveaux langages de programmation et de nouveaux concepts d'accès aux données.

Les requêtes SQL fédérées dans Athena éliminent cette complexité en permettant aux utilisateurs d'interroger les données sur place, où qu'elles se trouvent. Les analystes peuvent utiliser des structures SQL familières pour joindre (`JOIN`) des données à plusieurs sources de données pour une analyse rapide et stocker les résultats dans Simple Storage Service (Amazon S3) pour une utilisation ultérieure.

Connecteurs de source de données

Athena traite les requêtes fédérées à l'aide des connecteurs de sources de données Athena qui s'exécutent sur AWS Lambda. Utilisez ces connecteurs de source de données open source pour exécuter des requêtes SQL fédérées dans Athena sur Amazon DynamoDB, Apache HBase, Amazon Document DB, Amazon, CloudWatch CloudWatch Amazon Metrics et des bases de données relationnelles compatibles JDBC telles que MySQL et PostgreSQL sous licence Apache 2.0.

Connecteurs de source de données personnalisés

Grâce au kit Athena Query Federation SDK, les développeurs peuvent créer des connecteurs à n'importe quelle source de données pour permettre à Athena d'exécuter des requêtes SQL sur cette source de données. Athena Query Federation Connector étend les avantages des requêtes fédérées au-delà des connecteurs fournis. AWS Comme les connecteurs fonctionnent en AWS Lambda continu, vous n'avez pas à gérer l'infrastructure ni à planifier l'adaptation aux pics de demande.

Disponibilité de l'aperçu

La requête fédérée d’Athena est disponible en prévisualisation dans la région USA Est (Virginie du Nord).

Étapes suivantes

- Pour commencer votre prévisualisation, suivez les instructions de la FAQ des fonctions en prévisualisation d'Athena.
- Pour en savoir plus sur la fonction de requête fédérée, consultez la rubrique Utilisation de la requête fédérée d'Amazon Athena (prévisualisation).
- Pour commencer à utiliser un connecteur existant, reportez-vous à la section Déploiement d'un connecteur et connexion à une source de données.
- Pour savoir comment créer votre propre connecteur de source de données à l'aide du SDK Athena Query Federation, consultez Example Athena Connector on. GitHub

Invocation de modèles dde Machine Learning dans les requêtes SQL

Vous pouvez désormais invoquer des modèles de machine learning pour l'inférence directement à partir de vos requêtes Athena. La possibilité d'utiliser des modèles de machine learning dans les requêtes SQL rend les tâches complexes comme la détection d'anomalies, l'analyse de cohortes de clients, et les prédictions de ventes, aussi simples que l'invocation d'une fonction dans une requête SQL.

Modèles ML

Vous pouvez utiliser plus d'une douzaine d'algorithmes d'apprentissage automatique intégrés fournis par Amazon SageMaker, entraîner vos propres modèles ou rechercher des packages de modèles et les déployer sur Amazon SageMaker Hosting Services AWS Marketplaceet vous y abonner. Aucune configuration supplémentaire n'est requise. Vous pouvez invoquer ces modèles ML dans vos requêtes SQL à partir de la console Athena, des API Athena et du pilote JDBC de prévisualisation d'Athena.

Disponibilité de l'aperçu

La fonctionnalité ML d'Athena est disponible aujourd'hui en prévisualisation dans la région USA Est (Virginie du Nord).

Étapes suivantes

- Pour commencer votre prévisualisation, suivez les instructions de la FAQ des fonctions en prévisualisation d'Athena.
- Pour en savoir plus sur la fonction de machine learning, consultez la rubrique Utilisation de machine learning (ML) avec Amazon Athena (version de prévisualisation).

Fonctions définies par l'utilisateur (UDF) (version de prévisualisation)

Vous pouvez désormais écrire des fonctions scalaires personnalisées et les invoquer dans vos requêtes Athena. Vous pouvez écrire vos UDF en Java à l'aide du kit Athena Query Federation SDK. Lorsqu'une UDF est utilisée dans une requête SQL envoyée à Athena, elle est invoquée et exécutée sur AWS Lambda. Les UDF peuvent être utilisées à la fois dans `SELECT` et les clauses `FILTER` d'une requête SQL. Vous pouvez invoquer plusieurs UDF dans la même requête.

Disponibilité de l'aperçu

La fonctionnalité UDF d'Athena est disponible en mode Prévisualisation dans la région USA Est (Virginie du Nord).

Étapes suivantes

- Pour commencer votre prévisualisation, suivez les instructions de la FAQ des fonctions en prévisualisation d'Athena.
- Pour en savoir plus, consultez Interrogation avec des fonctions définies par l'utilisateur (version de prévisualisation).
- Pour des exemples d'implémentations UDF, consultez Amazon Athena UDF Connector activé. GitHub
- Pour apprendre à écrire vos propres fonctions à l'aide du kit Athena Query Federation SDK, consultez la rubrique Création et déploiement d'une UDF avec Lambda.

Utilisation du métastore Apache Hive comme métacatalogue avec Amazon Athena (version de prévisualisation)

Vous pouvez désormais connecter Athena à un ou plusieurs métastores Apache Hive en plus du AWS Glue Data Catalog avec Athena.

Connecteur Metastore

Pour vous connecter à un métastore Hive auto-hébergé, vous avez besoin d'un connecteur de métastore Hive Athena. Athena fournit un connecteur de mise en œuvre de référence que vous pouvez utiliser. Le connecteur fonctionne comme une AWS Lambda fonction de votre compte. Pour plus d'informations, consultez la rubrique Utilisation du connecteur de données Athena pour le métastore Hive externe (version de prévisualisation).

Disponibilité de l'aperçu

La fonction de métastore Hive est disponible en mode Prévisualisation dans la région USA Est (Virginie du Nord).

Étapes suivantes

- Pour commencer votre prévisualisation, suivez les instructions de la FAQ des fonctions en prévisualisation d'Athena.
- Pour en savoir plus sur cette fonction, veuillez consulter notre article intitulé Utilisation du connecteur de données Athena pour le métastore Hive externe (version de prévisualisation).

Nouvelles mesures liées à la requête

Athena publie désormais des métriques de requête supplémentaires qui peuvent vous aider à comprendre les performances d'Amazon Athena. Athena publie des statistiques relatives aux requêtes sur Amazon. CloudWatch Dans cette version, Athena publie les métriques de requête supplémentaires suivantes :

- Durée de planification de requêtes : temps nécessaire à la planification de la requête. Cela inclut le temps passé à récupérer les partitions de la table à partir de la source de données,
- Durée de mise en file d'attente des requêtes : temps pendant lequel la requête est restée dans une file d'attente de ressources.
- Durée de traitement du service : temps nécessaire à l'écriture des résultats après la fin du traitement du moteur de requête.
- Durée totale d'exécution : temps nécessaire pour qu'Athena exécute la requête.

Pour utiliser ces nouvelles métriques de requête, vous pouvez créer des tableaux de bord personnalisés, définir des alarmes et des déclencheurs sur les métriques ou utiliser des tableaux de bord préremplis directement depuis la console Athena. CloudWatch

Étapes suivantes

Pour plus d'informations, consultez la section Surveillance des requêtes Athena à l'aide CloudWatch de métriques.

12 novembre 2019

Date de publication : 17/12/2019

Amazon Athena est désormais disponible dans la région Moyen-Orient (Bahreïn).

8 novembre 2019

Date de publication : 17/12/2019

Amazon Athena est désormais disponible dans les régions USA Ouest (Californie du Nord) et Europe (Paris).

8 octobre 2019

Date de publication : 17/12/2019

Amazon Athena vous permet désormais de vous connecter directement à Athena via un point de terminaison de VPC d'interface dans votre cloud privé virtuel (VPC). Grâce à cette fonction, vous pouvez envoyer vos requêtes à Athena en toute sécurité sans avoir besoin d'une passerelle Internet dans votre VPC.

Pour créer un point de terminaison VPC d'interface pour vous connecter à Athena, vous pouvez utiliser le AWS Management Console ou (). AWS Command Line Interface AWS CLI Pour plus d'informations sur la création d'un point de terminaison d'interface, voir Création d'un point de terminaison d'interface.

Lorsque vous utilisez un point de terminaison VPC d'interface, la communication entre votre VPC et les API Athena est sécurisée et reste au sein du réseau. AWS Cette fonction est disponible sans frais supplémentaires pour Athena. Des frais s'appliquent pour le point de terminaison de VPC d'interface.

Pour en savoir plus sur cette fonction, consultez la rubrique Connexion à Amazon Athena à l'aide d'un point de terminaison de VPC d'interface.

19 septembre 2019

Date de publication : 17/12/2019

Amazon Athena ajoute la prise en charge de l'insertion de nouvelles données dans une table existante à l'aide de l'instruction `INSERT INTO`. Vous pouvez insérer de nouvelles lignes dans un

tableau de destination basé sur une instruction de requête `SELECT` qui s'exécute sur un tableau source, ou basé sur un ensemble de valeurs fourni dans le cadre d'une instruction de requête. Formats de données pris en charge : Avro, JSON, ORC, Parquet et fichiers textes.

Les instructions `INSERT INTO` peuvent également vous aider à simplifier votre processus ETL. Par exemple, vous pouvez utiliser `INSERT INTO` dans une seule requête pour sélectionner des données d'un tableau source au format JSON et écrire dans un tableau de destination au format Parquet.

Les instructions `INSERT INTO` sont facturées en fonction du nombre d'octets analysés dans la phase `SELECT`, de la même manière qu'Athena le fait pour les requêtes `SELECT`. Pour plus d'informations, consultez la rubrique Tarification Amazon Athena.

Pour plus d'informations sur l'utilisation`INSERT INTO`, y compris les formats pris en charge, SerDes et pour des exemples, voir INSERT INTO dans le guide de l'utilisateur d'Athena.

12 septembre 2019

Date de publication : 17/12/2019

Amazon Athena est désormais disponible dans la région Asie-Pacifique (Hong Kong).

16 août 2019

Date de publication : 17/12/2019

Amazon Athena ajoute la prise en charge de l'interrogation des données dans les compartiments Simple Storage Service (Amazon S3) de type Paiement par le demandeur.

Lorsqu'un compartiment Simple Storage Service (Amazon S3) est configuré en tant que Paiement par le demandeur, c'est le demandeur, et non le propriétaire du compartiment, qui paie la requête Simple Storage Service (Amazon S3) et les coûts de transfert des données. Dans Athena, les administrateurs de groupes de travail peuvent désormais configurer les paramètres des groupes de travail pour permettre aux membres de ces derniers d'interroger les compartiments S3 de type Paiement par le demandeur.

Pour plus d'informations sur la configuration du paramètre Paiement par le demandeur pour votre groupe de travail, reportez-vous à la section Création d'un groupe de travail du Guide de l'utilisateur d'Amazon Athena. Pour plus d'informations sur les compartiments de type Paiement par le demandeur, consultez la rubrique Compartiments de type Paiement par le demandeur du Guide du développeur Amazon Simple Storage Service.

9 août 2019

Date de publication : 17/12/2019

Amazon Athena prend désormais en charge l'application de politiques AWS Lake Formation pour le contrôle précis de l'accès à des bases de données, des tables et des colonnes, nouvelles ou existantes, définies dans le AWS Glue Data Catalog pour les données stockées dans Simple Storage Service (Amazon S3).

Vous pouvez utiliser cette fonctionnalité dans les pays suivants Régions AWS : USA Est (Ohio), USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo) et Europe (Irlande). Cette fonctionnalité est disponible sans frais additionnels.

Pour plus d'informations sur l'utilisation de cette fonction, consultez Utilisation d'Athena pour interroger des données enregistrées dans AWS Lake Formation. Pour plus d'informations sur AWS Lake Formation, consultez AWS Lake Formation.

26 juin 2019

Amazon Athena est désormais disponible dans la région Europe (Stockholm). Pour obtenir la liste des régions prises en charge, consultez Régions AWS et Points de terminaison.

24 mai 2019

Date de publication : 24/05/2019

Amazon Athena est désormais disponible dans les régions AWS GovCloud (USA Est) et AWS GovCloud (USA Ouest). Pour obtenir la liste des régions prises en charge, consultez Régions AWS et Points de terminaison.

5 mars 2019

Date de publication : 05/03/2019

Amazon Athena est désormais disponible dans la région Canada (Centre). Pour obtenir la liste des régions prises en charge, consultez Régions AWS et Points de terminaison. Publication de la nouvelle version du pilote ODBC avec prise en charge des groupes de travail Athena. Pour plus d'informations, consultez les Notes de mise à jour du pilote ODBC.

Pour télécharger le pilote ODBC version 1.0.5 et sa documentation, consultez Connexion à Amazon Athena avec le pilote ODBC. Pour plus d'informations sur cette version, consultez les Notes de mise à jour du pilote ODBC.

Pour utiliser des groupes de travail avec le pilote ODBC, définissez la nouvelle propriété de connexion, `Workgroup`, dans la chaîne de connexion, comme illustré dans l'exemple suivant :

```
Driver=Simba Athena ODBC
 Driver;AwsRegion=[Region];S3OutputLocation=[S3Path];AuthenticationType=IAM
 Credentials;UID=[YourAccessKey];PWD=[YourSecretKey];Workgroup=[WorkgroupName]
```

Pour plus d'informations, recherchez « groupe de travail » dans le Guide d'installation et de configuration du pilote ODBC version 1.0.5.Aucune modification apportée à la chaîne de connexion du pilote ODBC lorsque vous utilisez des identifications sur des groupes de travail. Pour utiliser des identifications, mettez à niveau vers la dernière version du pilote ODBC qui est la version actuelle.

Ce pilote vous permet d'utiliser des actions de groupe de travail d'API Athena pour créer et gérer des groupes de travail, et des actions d'étiquetage d'API Athena pour ajouter, répertorier ou supprimer des étiquettes sur les groupes de travail. Avant de commencer, veillez à disposer des autorisations au niveau des ressources dans IAM pour exécuter des actions sur les groupes de travail et des étiquettes.

Pour plus d'informations, voir :

- Utilisation de groupes de travail pour exécuter des requêtes et Exemples de politiques de groupe de travail.
- Étiquetage des ressources Athena et Politiques de contrôle d'accès IAM basées sur les étiquettes.

Si vous utilisez le pilote JDBC ou le AWS SDK, passez à la dernière version du pilote et du SDK, qui incluent déjà la prise en charge des groupes de travail et des balises dans Athena. Pour plus d'informations, consultez Connexion à Amazon Athena avec JDBC.

22 février 2019

Date de publication : 22/02/2019

Ajout de la prise en charge des étiquettes pour les groupes de travail dans Amazon Athena. une identification est constituée d'une clé et d'une valeur que vous définissez. Lorsque vous identifiez un groupe de travail, vous lui attribuez des métadonnées personnalisées. Vous pouvez ajouter des

balises aux groupes de travail pour les classer par catégories, en utilisant les meilleures pratiques en matière de AWS balisage. Vous pouvez utiliser des identifications pour limiter l'accès aux groupes de travail et pour suivre les coûts. Par exemple, créez un groupe de travail pour chaque centre de coûts. Ensuite, en ajoutant des étiquettes à ces groupes de travail, vous pouvez suivre vos dépenses Athena pour chaque centre de coûts. Pour plus d'informations, consultez Utilisation d'identifications pour la facturation dans le guide de l'utilisateur AWS Billing and Cost Management .

Vous pouvez travailler avec des étiquettes en utilisant la console Athena ou les opérations d'API. Pour plus d'informations, consultez Étiquetage des ressources Athena.

Dans la console Athena, vous pouvez ajouter une ou plusieurs étiquettes à chacun de vos groupes de travail et effectuer une recherche par étiquette. Les groupes de travail sont une ressource contrôlée par IAM dans Athena. Dans IAM, vous pouvez limiter les personnes autorisées à ajouter, supprimer ou répertorier des étiquettes sur des groupes de travail que vous créez. Vous pouvez également utiliser l'opération d'API `CreateWorkGroup` possédant le paramètre d'identification facultative pour ajouter une ou plusieurs identifications au groupe de travail. Pour ajouter, supprimer ou répertorier des identifications, utilisez `TagResource`, `UntagResource` et `ListTagsForResource`. Pour plus d'informations, consultez Utilisation des opérations d'identification.

Pour permettre aux utilisateurs d'ajouter des étiquettes lors de la création de groupes de travail, veillez à accorder des autorisations IAM à chaque utilisateur pour exécuter les actions d'API `TagResource` et `CreateWorkGroup`. Pour plus d'informations et d'exemples, consultez Politiques de contrôle d'accès IAM basées sur les étiquettes.

Aucune modification apportée au pilote JDBC lorsque vous utilisez des identifications sur des groupes de travail. Si vous créez de nouveaux groupes de travail et utilisez le pilote JDBC ou le AWS SDK, passez à la dernière version du pilote et du SDK. Pour plus d'informations, veuillez consulter Connexion à Amazon Athena avec JDBC.

18 février 2019

Date de publication : 18/02/2019

Ajout de la possibilité de contrôler les coûts de requête en exécutant des requêtes dans des groupes de travail. Pour plus d'informations, veuillez consulter Utilisation des groupes de travail pour contrôler l'accès aux requêtes et les coûts. Amélioration du JSON OpenX SerDe utilisé dans Athena, correction d'un problème en raison duquel Athena n'ignorait pas les objets transférés vers la classe de `GLACIER` stockage et ajout d'exemples d'interrogation des journaux Network Load Balancer.

Modifications suivantes effectuées :

- Ajout de la prise en charge des groupes de travail. Utilisation de groupes de travail pour séparer les utilisateurs, les équipes, les applications ou les charges de travail, et pour définir des limites au volume de données pouvant être traité par chaque requête ou groupe de travail entier. Vous pouvez utiliser des autorisations au niveau des ressources IAM pour contrôler l'accès à un groupe de travail spécifique, car les groupes de travail agissent en tant que ressources IAM. Vous pouvez également consulter les métriques relatives aux requêtes dans Amazon CloudWatch, contrôler les coûts des requêtes en limitant la quantité de données numérisées, créer des seuils et déclencher des actions, telles que des alarmes Amazon SNS, lorsque ces seuils sont dépassés. Pour plus d'informations, consultez Utilisation de groupes de travail pour exécuter des requêtes et Contrôle des coûts et surveillance des requêtes à l'aide de CloudWatch métriques et d'événements.

  Les groupes de travail sont une ressource IAM. Pour une liste complète des actions, ressources et conditions liées aux groupes de travail dans IAM, consultez la rubrique Actions, ressources et clés de condition pour Amazon Athena dans la Référence d'autorisation de service. Avant de créer de nouveaux groupes de travail, assurez-vous que vous utilisez des politiques IAM de groupe de travail et la AWS politique gérée : AmazonAthenaFullAccess.

  Vous pouvez utiliser des groupes de travail dans la console avec les opérations d'API de groupe de travail ou avec le pilote JDBC. Pour une procédure de haut niveau, consultez Configuration de groupes de travail. Pour télécharger le pilote JDBC avec prise en charge de groupe de travail, consultez Connexion à Amazon Athena avec JDBC.

  Si vous utilisez des groupes de travail avec le pilote JDBC, vous devez définir le nom du groupe de travail dans la chaîne de connexion à l'aide du paramètre de configuration `Workgroup`, comme illustré dans l'exemple suivant :

  ```
  jdbc:awsathena://AwsRegion=<AWSREGION>;UID=<ACCESSKEY>;
  PWD=<SECRETKEY>;S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/<athena-
  output>-<AWSREGION>/;
  Workgroup=<WORKGROUPNAME>;
  ```

  Aucune modification dans la manière d'exécuter des instructions SQL ou d'effectuer des appels d'API JDBC au pilote. Le pilote transmet le nom du groupe de travail à Athena.

  Pour obtenir des informations sur les différences introduites avec les groupes de travail, consultez API de groupes de travail Athena et Dépannage des groupes de travail.

- Amélioration du JSON OpenX SerDe utilisé dans Athena. Ces améliorations incluent, sans toutefois s'y limiter :
  - Prise en charge de la propriété `ConvertDotsInJsonKeysToUnderscores`. Lorsqu'il est défini sur`TRUE`, il permet de SerDe remplacer les points dans les noms clés par des traits de soulignement. Par exemple, si le jeu de données JSON contient une clé portant le nom `"a.b"`, vous pouvez utiliser cette propriété pour définir le nom de la colonne comme étant `"a_b"` dans Athena. L'argument par défaut est `FALSE`. Par défaut, Athena n'autorise pas les points dans les noms de colonnes.
  - Prise en charge de la propriété `case.insensitive`. Par défaut, Athena exige que toutes les clés de votre jeu de données JSON soient en minuscules. `WITH SERDE PROPERTIES ("case.insensitive"= FALSE;)` vous permet d'utiliser des noms de clé sensibles à la casse dans vos données. L'argument par défaut est `TRUE`. Lorsqu'il est défini sur`TRUE`, il SerDe convertit toutes les colonnes majuscules en minuscules.

  Pour plus d'informations, consultez OpenX JSON SerDe.
- Correction d'un problème à cause duquel le service Athena renvoyait des messages d'erreur `"access denied"` lorsqu'il traitait des objets Simple Storage Service (Amazon S3) archivés dans Glacier par des politiques de cycle de vie Simple Storage Service (Amazon S3). Suite à la correction de ce problème, Athena ignore les objets passés à la classe de stockage `GLACIER`. Athena ne prend pas en charge l'interrogation des données à partir de la classe de stockage `GLACIER`.

  Pour plus d'informations, veuillez consulter les rubriques the section called “Exigences pour les tables dans Athena et les données dans Simple Storage Service (Amazon S3)” et Transition vers la classe de stockage GLACIER (archivage d'objets) du Guide de l'utilisateur Amazon Simple Storage Service.
- Ajout d'exemples d'interrogation des journaux d'accès du Network Load Balancer qui reçoivent des informations sur les requêtes TLS (Transport Layer Security, Sécurité de la couche de transport). Pour plus d'informations, consultez the section called “Network Load Balancer”.

Notes de publication d'Athena pour 2018

20 novembre 2018

Date de publication : 20/11/2018

Lancement des nouvelles versions des pilotes JDBC et ODBC avec prise en charge de l'accès fédéré à l'API Athena avec AD FS et SAML 2.0 (Security Assertion Markup Language 2.0). Pour plus de détails, consultez les Notes de mise à jour du pilote JDBC et les Notes de mise à jour du pilote ODBC.

Avec cette version, l'accès fédéré à Athena est pris en charge pour Active Directory Federation Service (AD FS 3.0). L'accès est établi via les versions des pilotes JDBC ou ODBC prenant en charge SAML 2.0. Pour en savoir plus sur la configuration de l'accès fédéré à l'API Athena, voir the section called “Activation de l'accès fédéré à l'API Athena”.

Pour télécharger le pilote JDBC version 2.0.6 et sa documentation, consultez Connexion à Amazon Athena avec JDBC. Pour plus d'informations sur cette version, consultez les Notes de mise à jour du pilote JDBC.

Pour télécharger le pilote ODBC version 1.0.4 et sa documentation, consultez Connexion à Amazon Athena avec le pilote ODBC. Pour plus d'informations sur cette version, consultez les Notes de mise à jour du pilote ODBC.

Pour plus d'informations sur la prise en charge de SAML 2.0 dans AWS, voir À propos de la fédération SAML 2.0 dans le guide de l'utilisateur IAM.

15 octobre 2018

Date de publication : 15/10/2018

Si vous avez effectué la mise à niveau vers le AWS Glue Data Catalog, deux nouvelles fonctionnalités permettent de prendre en charge les éléments suivants :

- Chiffrement des métadonnées du catalogue de données. Si vous choisissez de chiffrer les métadonnées dans le catalogue de données, vous devez ajouter des politiques spécifiques à Athena. Pour en savoir plus, consultez Accès aux métadonnées chiffrées dans le AWS Glue Data Catalog.
- Autorisations précises pour accéder aux ressources du. AWS Glue Data Catalog Vous pouvez désormais définir des politiques basées sur l'identité (IAM) qui restreignent ou autorisent l'accès à des bases de données et des tables spécifiques à partir du catalogue de données utilisé dans Athena. Pour plus d’informations, consultez Accès détaillé aux bases de données et aux tables du AWS Glue Data Catalog.

> **Note**
>
> Les données résident dans les compartiments Amazon S3 et leur accès est contrôlé par Accès à Amazon S3 depuis Athena. Pour accéder aux données des bases de données et des tables, continuez à utiliser des politiques de contrôle d'accès aux compartiments Simple Storage Service (Amazon S3) qui stockent les données.

10 octobre 2018

Date de publication : 10/10/2018

Athena prend en charge `CREATE TABLE AS SELECT`, ce qui crée une table à partir du résultat d'une instruction de requête `SELECT`. Pour plus de détails, consultez la section Création d'une table à partir des résultats des requêtes (CTAS).

Avant de créer des requêtes CTAS, il est important d'en savoir plus sur leur comportement dans la documentation Athena. Elle contient des informations sur l'emplacement pour enregistrer les résultats de requête dans Simple Storage Service (Amazon S3), la liste des formats pris en charge pour stocker les résultats de requête CTAS, le nombre de partitions que vous pouvez créer et les formats de compression pris en charge. Pour plus d'informations, consultez Considérations et limitations relatives aux requêtes CTAS.

Utilisez les requêtes CTAS pour :

- Créez une table à partir des résultats de la requête en une étape.
- Créez des requêtes CTAS dans la console Athena, à l'aide d'exemples. Pour obtenir des informations sur la syntaxe, consultez CREATE TABLE AS.
- Transformez les résultats des requêtes en d'autres formats de stockage, tels que PARQUET, ORC, AVRO, JSON et TEXTFILE. Pour plus d'informations, consultez Considérations et limitations relatives aux requêtes CTAS et Formats de stockage en colonnes.

6 septembre 2018

Date de publication : 06/09/2018

Publication de la nouvelle version du pilote ODBC (version 1.0.3). La nouvelle version du pilote ODBC diffuse les résultats par défaut, au lieu de les paginer, ce qui permet aux outils de business

intelligence de récupérer de grands ensembles de données plus rapidement. Cette version inclut également des améliorations, des correctifs de bogues et une mise à jour de la documentation pour « Utilisation de SSL avec un serveur proxy ». Pour plus de détails, consultez les Notes de mise à jour du pilote.

Pour plus d'informations sur le téléchargement du pilote ODBC version 1.0.3 et de sa documentation, consultez Connexion à Amazon Athena avec le pilote ODBC.

La fonction de streaming des résultats est uniquement disponible avec cette nouvelle version du pilote ODBC. Elle est également disponible avec le pilote JDBC. Pour plus d'informations sur les résultats du streaming, consultez le Guide d'installation et de configuration du pilote ODBC et recherchez UseResultsetStreaming.

Le pilote ODBC version 1.0.3 remplace la version précédente du pilote. Nous vous recommandons de migrer vers le pilote en cours.

> **Important**
>
> Pour utiliser le pilote ODBC version 1.0.3, suivez ces exigences :
>
> - Gardez le port 444 ouvert pour le trafic sortant.
> - Ajoutez l'action de politique `athena:GetQueryResultsStream` à la liste des politiques pour Athena. Cette action de politique n'est pas exposée directement avec l'API et est utilisé uniquement avec les pilotes ODBC et JDBC, dans le cadre de la prise en charge des résultats de streaming. Pour un exemple de politique, consultez AWS politique gérée : AWSQuicksightAthenaAccess.

23 août 2018

Date de publication : 23/08/2018

Ajout de la prise en charge de ces fonctionnalités liées à DDL et correction de plusieurs bogues, comme suit :

- Ajout de la prise en charge pour les types de données `BINARY` et `DATE` des données dans Parquet, et pour les types de données `DATE` et `TIMESTAMP` pour les données dans Avro.
- Ajout de la prise en charge de `INT` et `DOUBLE` dans les requêtes DDL. `INTEGER` est un alias de `INT` et `DOUBLE PRECISION` un alias de `DOUBLE`.

- Amélioration des performances des requêtes `DROP TABLE` et `DROP DATABASE`.
- Suppression de la création d'un objet `_$folder$` dans Simple Storage Service (Amazon S3) lorsqu'un compartiment de données est vide.
- Résolution d'un problème où `ALTER TABLE ADD PARTITION` génère une erreur quand aucune valeur de partition n'a été fournie.
- Résolution d'un problème où `DROP TABLE` a ignoré le nom de base de données lors de la vérification des partitions après que le nom qualifié a été spécifié dans l'instruction.

Pour plus d'informations sur les types de données prises en charge dans Athena, consultez Types de données dans Amazon Athena.

Pour en savoir plus sur les types de mappages entre des types de données pris en charge dans Athena, le pilote JDBC et les types de données Java, consultez la section « Types de données » du Guide de configuration et d'installation du pilote JDBC.

16 août 2018

Date de publication : 16/08/2018

Publication du pilote JDBC version 2.0.5. La nouvelle version du pilote JDBC diffuse les résultats par défaut, au lieu de les paginer, ce qui permet aux outils de business intelligence de récupérer de grands ensembles de données plus rapidement. Par rapport à la version précédente du pilote JDBC, il y a les améliorations de performances suivantes :

- Augmentation des performances de 2 fois environ lors de l'extraction des performances de moins de 10 000 lignes.
- Augmentation des performances de 5 à 6 fois environ lors de l'extraction des performances de plus de 10 000 lignes.

La fonction de streaming des résultats est uniquement disponible avec le pilote JDBC. Elle n'est pas disponible avec le pilote ODBC. Vous ne pouvez pas l'utiliser avec l'API Athena. Pour plus d'informations sur les résultats du streaming, consultez le Guide d'installation et de configuration du pilote JDBC et recherchez. UseResultsetStreaming

Pour plus d'informations sur le téléchargement du pilote JDBC version 2.0.5 et de sa documentation, consultez Connexion à Amazon Athena avec JDBC.

Le pilote JDBC version 2.0.5 remplace la version précédente du pilote (2.0.2). Pour vous assurer que vous pouvez utiliser le pilote JDBC en version 2.0.5, ajoutez la politique d'action `athena:GetQueryResultsStream` à la liste des politiques pour Athena. Cette action de politique n'est pas exposée directement avec l'API et est utilisé uniquement avec le pilote JDBC, dans le cadre de la prise en charge des résultats de streaming. Pour un exemple de politique, consultez AWS politique gérée : AWSQuicksightAthenaAccess. Pour plus d'informations sur la migration vers la version 2.0.2 depuis la version 2.0.5 du pilote, consultez le Guide de migration du pilote JDBC.

Si vous effectuez une migration depuis un pilote 1.x vers un pilote 2.x, vous devrez migrer vos configurations existantes vers la nouvelle configuration. Nous vous recommandons vivement de migrer vers la version courante du pilote. Pour plus d'informations, consultez le Guide de la migration du pilote JDBC.

7 août 2018

Date de publication : 07/08/2018

Vous pouvez désormais stocker les journaux de flux du cloud privé virtuel d'Amazon directement dans Simple Storage Service (Amazon S3) au format GZIP, où vous pouvez les interroger dans Athena. Pour obtenir des informations, consultez Interrogation des journaux de flux Amazon VPC et . Les journaux de flux Amazon VPC peuvent désormais être diffusés vers S3.

5 juin 2018

Date de publication : 05/06/2018

Rubriques

- Prise en charge des vues
- Améliorations et mises à jour des messages d'erreur
- Correctifs de bogue

Prise en charge des vues

Ajout de la prise en charge des vues. Vous pouvez désormais utiliser CREATE VIEW, DESCRIBE VIEW, DROP VIEW, SHOW CREATE VIEW et SHOW VIEWS dans Athena. La requête qui définit la vue est exécutée chaque fois que vous référencez la vue dans votre requête. Pour plus d'informations, consultez Utilisation des vues.

Améliorations et mises à jour des messages d'erreur

- Une bibliothèque GSON 2.8.0 a été incluse dans le CloudTrail SerDe, afin de résoudre un problème lié à l'analyse des chaînes JSON CloudTrail SerDe et de permettre leur analyse.
- Amélioration de la validation du schéma de partition dans Athena pour Parquet et, dans certains cas, pour ORC, en permettant la réorganisation des colonnes. Cela permet à Athena de mieux gérer les modifications de l'évolution du schéma au fil du temps, ainsi que les tables ajoutées par le AWS Glue Crawler. Pour plus d'informations, consultez Traitement des mises à jour de schéma.
- Ajout de la prise en charge de l'analyse pour `SHOW VIEWS`.
- Améliorations suivantes apportées à la plupart des messages d'erreur courants :
  - Un message d'erreur interne a été remplacé par un message d'erreur descriptif en cas d' SerDe échec de l'analyse de la colonne dans une requête Athena. Auparavant, Athena émettait une erreur interne en cas d'erreurs d'analyse. Le nouveau message d'erreur indique : « HIVE_BAD_DATA: Error parsing field value for field 0: java.lang.String cannot be cast to org.openx.data.jsonserde.json.JSONObject ».
  - Amélioration des messages d'erreur concernant des autorisations insuffisantes par l'ajout de détails.

Correctifs de bogue

Les bogues suivants ont été corrigés :

- Résolution d'un problème qui permet la conversion de `REAL` en types de données `FLOAT`. Cela améliore l'intégration au Crawler AWS Glue qui renvoie les types de données `FLOAT`.
- Correction d'un problème où Athena ne convertissait pas AVRO `DECIMAL` (un type logique) en un type `DECIMAL`.
- Correction d'un problème pour lequel Athena ne renvoyait pas les résultats des requêtes sur les données Parquet avec des clauses `WHERE` faisant référence à des valeurs dans le type de données `TIMESTAMP`.

17 mai 2018

Date de publication : 17/05/2018

Augmentation des quotas de simultanéité des requêtes dans Athena de cinq à vingt. Cela signifie que vous pouvez soumettre et exécuter jusqu'à vingt requêtes `DDL` et vingt requêtes `SELECT` en même temps. Notez que les quotas de simultanéité sont distincts pour les requêtes `DDL` et `SELECT`.

Les quotas de simultanéité dans Athena sont définis en tant que nombre de requêtes pouvant être soumises au service simultanément. Vous pouvez soumettre jusqu'à vingt requêtes du même type (`DDL` or `SELECT`) en même temps. Si vous soumettez une requête dépassant le quota de requêtes simultanées, l'API Athena affiche un message d'erreur.

Une fois vos requêtes soumises à Athena, celui-ci traite les requêtes en affectant des ressources en fonction de la charge de service globale et du volume de demandes entrantes. Nous surveillons et apportons en continu les ajustements de service afin que le traitement de vos requêtes soit aussi rapide que possible.

Pour plus d'informations, veuillez consulter Service Quotas. Il s'agit d'un quota ajustable. Vous pouvez utiliser la console Service Quotas pour demander une augmentation du quota de requêtes simultanées.

19 avril 2018

Date de publication : 19/04/2018

Publication de la nouvelle version du pilote JDBC (version 2.0.2) avec prise en charge du renvoi de données `ResultSet` en tant que type de données Tableau, améliorations et correctifs de bogue. Pour plus de détails, consultez les Notes de mise à jour du pilote.

Pour plus d'informations sur le téléchargement du nouveau pilote JDBC version 2.0.2 et de sa documentation, consultez Connexion à Amazon Athena avec JDBC.

La version la plus récente du pilote JDBC est la version 2.0.2. Si vous effectuez une migration depuis un pilote 1.x vers un pilote 2.x, vous devrez migrer vos configurations existantes vers la nouvelle configuration. Nous vous recommandons vivement de migrer vers le pilote en cours.

Pour plus d'informations sur les changements introduits dans la nouvelle version du pilote, les différences de version, et des exemples, consultez la section JDBC Driver Migration Guide (Guide de migration du pilote JDBC).

6 avril 2018

Date de publication : 06/04/2018

Utilisation de la saisie semi-automatique pour saisir des requêtes dans la console Athena.

15 mars 2018

Date de publication : 15/03/2018

Ajout de la possibilité de créer automatiquement des tables Athena pour les fichiers CloudTrail journaux directement depuis la CloudTrail console. Pour plus d'informations, veuillez consulter Utilisation de la CloudTrail console pour créer une table Athena pour les journaux CloudTrail .

2 février 2018

Date de publication : 12/02/2018

Ajout de la possibilité de décharger en toute sécurité des données intermédiaires sur le disque pour les requêtes nécessitant beaucoup de mémoire qui utilisent la clause `GROUP BY`. Cela permet d'améliorer la fiabilité de ces requêtes et empêche les erreurs liées à l'épuisement des ressources de requête.

19 janvier 2018

Date de publication : 19/01/2018

Athena utilise Presto, un moteur de requête open source, pour exécuter des requêtes.

Avec Athena, il n'y a pas de versions à gérer. Nous avons mis à niveau de façon transparente le moteur sous-jacent dans Athena vers une version basée sur Presto version 0.172. Aucune action de votre part n'est nécessaire.

Grâce à la mise à niveau, vous pouvez désormais utiliser les fonctions et opérateurs Presto 0.172, y compris les expressions Lambda Presto 0.172 dans Athena.

Les mises à jour majeures de cette version, y compris les corrections développées par la communauté, incluent :

- Prise en charge du non-respect des en-têtes. Vous pouvez utiliser la propriété `skip.header.line.count` lors de la définition de tables pour autoriser Athena à ignorer les en-têtes. Ceci est pris en charge pour les requêtes qui utilisent SerDeOpenCSV LazySimpleSerDeet non pour Grok ou Regex. SerDes
- Prise en charge du type de données `CHAR(n)` dans les fonctions `STRING`. La plage pour `CHAR(n)` est `[1.255]`, tandis que la plage pour `VARCHAR(n)` est `[1,65535]`.

- Prise en charge des sous-requêtes corrélées.
- Prise en charge des expressions et fonctions lambda Presto.
- Amélioration des performances du type `DECIMAL` et des opérateurs.
- Prise en charge des agrégations filtrées, telles que `SELECT sum(col_name) FILTER`, où `id > 0`.
- Déploiement des prédicats pour les types de données `DECIMAL`, `TINYINT`, `SMALLINT` et `REAL`.
- Prise en charge des prédicats de comparaison quantifiée : `ALL`, `ANY` et `SOME`.
- Ajout des fonctions : `arrays_overlap()`, `array_except()`, `levenshtein_distance()`, `codepoint()`, `skewness()`, `kurtosis()` et `typeof()`.
- Ajout d'une variante de la fonction `from_unixtime()` qui accepte un argument de fuseau horaire.
- Ajout des fonctions d'agrégation `bitwise_and_agg()` et `bitwise_or_agg()`.
- Ajout des fonctions `xxhash64()` et `to_big_endian_64()`.
- Ajout de la prise en charge de l'échappement des guillemets doubles et des barres obliques inverses à l'aide d'une barre oblique inverse et d'un indice de chemin JSON vers les fonctions `json_extract()` et `json_extract_scalar()`. Cela change la sémantique de toute invocation utilisant une barre oblique inverse, étant donné que les barres obliques inverses étaient précédemment considérées comme des caractères normaux.

Pour plus d'informations sur les fonctions et les opérateurs, voir Requêtes, fonctions et opérateurs DML dans ce guide et Fonctions et opérateurs dans la documentation Presto.

Athena ne prend pas en charge toutes les fonctions Presto. Pour plus d’informations, consultez Limites.

Notes de publication d'Athena pour 2017

13 novembre 2017

Date de publication : 13/11/2017

Ajout de la prise en charge de la connexion d'Athena au pilote ODBC. Pour plus d’informations, veuillez consulter Connexion à Amazon Athena avec le pilote ODBC.

1 novembre 2017

Date de publication : 01/11/2017

Ajout de la prise en charge pour les requêtes de données géospatiales, et pour les régions Asie-Pacifique (Séoul), Asie-Pacifique (Mumbai) et UE (Londres). Pour plus d'informations, consultez Interrogation de données géospatiales, Régions AWS et Points de terminaison.

19 octobre 2017

Date de publication : 19/10/2017

Ajout de la prise en charge pour UE (Francfort). Pour accéder à la liste des régions prises en charge, consultez Régions AWS et Points de terminaison.

3 octobre 2017

Date de publication : 03/10/2017

Créez des requêtes Athena nommées avec. AWS CloudFormation Pour plus d'informations, consultez AWS::Athena::NamedQueryle guide de AWS CloudFormation l'utilisateur.

25 septembre 2017

Date de publication : 25/09/2017

Ajout de la prise en charge de l'Asie-Pacifique (Sydney). Pour accéder à la liste des régions prises en charge, consultez Régions AWS et Points de terminaison.

14 août 2017

Date de publication : 14/08/2017

Intégration ajoutée avec le AWS Glue Data Catalog et assistant de migration pour la mise à jour du catalogue de données géré Athena vers le. AWS Glue Data Catalog Pour plus d'informations, consultez Intégration avec AWS Glue.

4 août 2017

Date de publication : 04/08/2017

Ajout de la prise en charge de Grok SerDe, qui facilite la correspondance de modèles pour les enregistrements dans des fichiers texte non structurés tels que les journaux. Pour plus d'informations, consultez Grok SerDe. Ajout de raccourcis clavier pour faire défiler l'historique des requêtes à l'aide de la console (CTRL+⇧/⇩ dans Windows, CMD+⇧/⇩ sur Mac).

22 juin 2017

Date de publication : 22/06/2017

Ajout de la prise en charge des régions Asie-Pacifique (Tokyo) et Asie-Pacifique (Singapour). Pour accéder à la liste des régions prises en charge, consultez Régions AWS et Points de terminaison.

8 juin 2017

Date de publication : 08/06/2017

Ajout de la prise en charge de l'Europe (Irlande). Pour plus d'informations, consultez Régions AWS and Endpoints.

19 mai 2017

Date de publication : 19/05/2017

Ajout d'une API Amazon Athena et AWS CLI prise en charge d'Athena ; mise à jour du pilote JDBC vers la version 1.1.0 ; résolution de divers problèmes.

- Amazon Athena permet la programmation d'application pour Athena. Pour plus d'informations, consultez la Référence d'API Amazon Athena. Les derniers AWS SDK incluent la prise en charge de l'API Athena. Pour des liens vers la documentation et les téléchargements, consultez la section SDK dans Outils pour Amazon Web Services.
- AWS CLI Cela inclut de nouvelles commandes pour Athéna. Pour plus d'informations, consultez la rubrique Référence d'API Amazon Athena.
- Un nouveau pilote JDBC 1.1.0 est disponible, qui prend en charge la nouvelle API Athena ainsi que les dernières fonctionnalités et corrections de bogues. Téléchargez le pilote à l'adresse https://downloads.athena.us-east-1.amazonaws.com/drivers/AthenaJDBC41-1.1.0.jar. Nous vous recommandons d'effectuer la mise à niveau vers la dernière version du pilote JDBC d'Athena. Toutefois, vous pouvez encore utiliser l'ancienne version du pilote. Les versions antérieures du pilote ne prennent pas en charge l'API Athena. Pour plus d'informations, consultez Connexion à Amazon Athena avec JDBC.
- Les actions spécifiques aux déclarations de politique dans les versions antérieures d'Athena sont désormais obsolètes. Si vous effectuez une mise à niveau vers la version 1.1.0 du pilote JDBC et avez des politiques IAM en ligne ou gérées par le client, associées aux utilisateurs JDBC, vous

devez mettre à jour les politiques IAM. En revanche, les versions antérieures du pilote JDBC ne prennent pas en charge l'API Athena, si bien que vous pouvez spécifier uniquement des actions obsolètes dans les politiques associées aux utilisateurs d'une version antérieure de JDBC. C'est pourquoi vous ne devriez pas avoir besoin de mettre à jour les politiques IAM en ligne ou gérées par le client.

- Ces actions spécifiques de politique ont été utilisées dans Athena avant la parution de l'API Athena. Utilisez ces actions obsolètes dans les politiques uniquement avec les pilotes JDBC antérieurs à la version 1.1.0. Si vous mettez à niveau le pilote JDBC, remplacez les déclarations de politique qui autorisent ou refusent les actions obsolètes par les actions d'API appropriées telles que listées, sinon des erreurs se produiront.

Action spécifique de politique obsolète	Action d'API Athena correspondante
`athena:RunQuery`	`athena:StartQueryExecution`
`athena:CancelQueryExecution`	`athena:StopQueryExecution`
`athena:GetQueryExecutions`	`athena:ListQueryExecutions`

Améliorations

- Augmentation de la longueur limite des chaînes de requête à 256 Ko.

Correctifs de bogue

- Correction d'un problème selon lequel des résultats de requête semblaient incorrects lorsque vous les faisiez défiler dans la console.
- Correction d'un problème selon lequel une chaîne de caractères `\u0000` dans des fichiers de données Simple Storage Service (Amazon S3) entraînait des erreurs.
- Correction d'un problème qui provoquait l'échec des demandes d'annulation d'une requête effectuée via le pilote JDBC.
- Correction d'un problème qui provoquait AWS CloudTrail SerDe l'échec des données Amazon S3 dans l'est des États-Unis (Ohio).

- Résolution d'un problème lié à l'échec de `DROP TABLE` sur une table partitionnée.

4 avril 2017

Date de publication : 04/04/2017

Ajout de la prise en charge du chiffrement des données Simple Storage Service (Amazon S3) et publication de la mise à jour du pilote JDBC (version 1.0.1) avec des améliorations de la prise en charge du chiffrement et des corrections de bogues.

Fonctionnalités

- Les fonctionnalités de chiffrement suivantes ont été ajoutées :
  - Prise en charge des requêtes de données chiffrées dans Simple Storage Service (Amazon S3).
  - Prise en charge du chiffrement des résultats de requête Athena.
- Une nouvelle version du pilote prend en charge les nouvelles fonctions de chiffrement, ajoute des améliorations et corrige des bogues.
- Ajout de la possibilité d'ajouter, de remplacer et de modifier des colonnes avec `ALTER TABLE`. Pour plus d'informations, consultez Alter Column dans la documentation Hive.
- Ajout de la prise en charge des requêtes de données compressées par LZO.

Pour plus d’informations, consultez Chiffrement au repos.

Améliorations

- Meilleures performances des requêtes JDBC avec une taille de page améliorée, renvoyant 1 000 lignes au lieu de 100.
- Ajout de la possibilité d'annuler une requête à l'aide de l'interface du pilote JDBC.
- Ajout de la possibilité de spécifier des options JDBC dans l'URL de connexion JDBC. Consultez Connexion à Amazon Athena avec JDBC pour obtenir le pilote JDBC le plus récent.
- Ajout du paramètre PROXY dans le pilote, qui peut désormais être défini ClientConfigurationdans le AWS SDK for Java.

Correctifs de bogue

Les bogues suivants ont été corrigés :

- Des erreurs de limitation pouvaient se produire lorsque plusieurs requêtes étaient émises via l'interface du pilote JDBC.
- Le pilote JDBC s'interrompait lors de la projection d'un type de données décimal.
- Le pilote JDBC renvoyait chaque type de données sous la forme d'une chaîne, quelle qu'était la façon dont le type de données était défini dans la table. Par exemple, la sélection d'une colonne définie comme type de données `INT` en utilisant `resultSet.GetObject()` renvoyait un type de données `STRING` à la place d'un type `INT`.
- Le pilote JDBC vérifiait les informations d'identification au moment où une connexion était effectuée, plutôt qu'au moment où une requête était exécutée.
- Les requêtes effectuées via le pilote JDBC échouaient lorsqu'un schéma était spécifié avec l'URL.

24 mars 2017

Date de publication : 24/03/2017

Ajout de l' AWS CloudTrail SerDeamélioration des performances, résolution des problèmes de partition.

Fonctionnalités

- Ajouté le AWS CloudTrail SerDe, qui a depuis été remplacé par le Hive JSON SerDe pour lire CloudTrail les journaux. Pour plus d'informations sur l'interrogation CloudTrail des journaux, consultezJournaux d'interrogation AWS CloudTrail.

Améliorations

- Amélioration des performances lors de l'analyse d'un grand nombre de partitions.
- Amélioration des performances sur l'opération `MSCK Repair Table`.
- Ajout de la possibilité d'exécuter des requêtes sur les données Simple Storage Service (Amazon S3) stockées dans des régions autres que votre région principale. Les taux standard de transfert de données entre régions pour Simple Storage Service (Amazon S3) s'appliquent en plus des frais Athena standard.

Correctifs de bogue

- Correction d'un bogue qui entraînait éventuellement une erreur de type « table introuvable » si aucune partition n'était chargée.
- Correction d'un bogue pour éviter de lever une exception avec les requêtes `ALTER TABLE ADD PARTITION IF NOT EXISTS`.
- Correction d'un bogue dans `DROP PARTITIONS`.

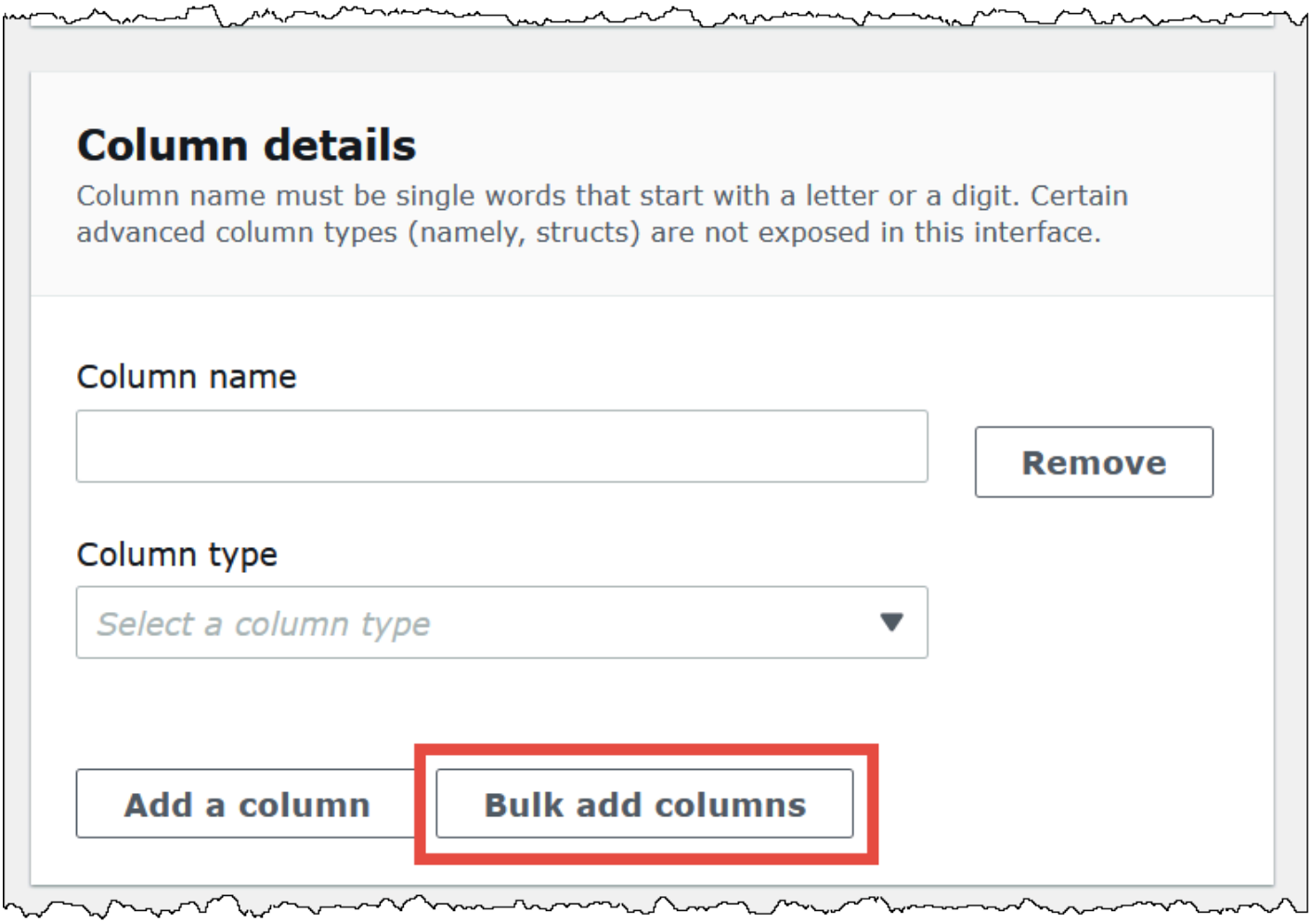
20 février 2017

Date de publication : 20/02/2017

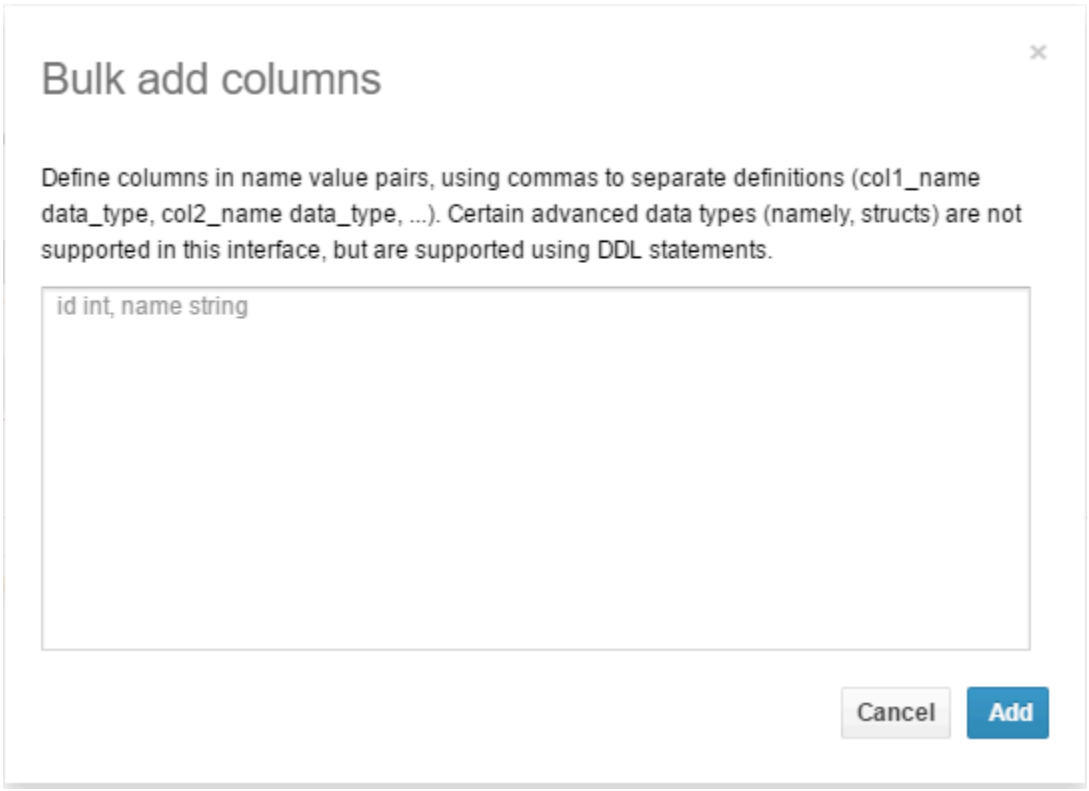
Ajout de la prise en charge AvroSerDe d'SerDeOpenCSV, de la région USA Est (Ohio) et de l'édition groupée de colonnes dans l'assistant de console. Amélioration des performances sur les tables Parquet volumineuses.

Fonctionnalités

- Support introduit pour les nouveaux SerDes :
  - SerDe Avro
  - SerDe OpenCSV pour le traitement des fichiers CSV
- Lancement de la région USA Est (Ohio) (us-east-2). Vous pouvez désormais exécuter des requêtes dans cette région.
- Vous pouvez désormais utiliser le formulaire Create Table From S3 bucket data (Créer une table à partir des données du compartiment S3) pour définir le schéma de table en bloc. Dans l'éditeur de requêtes, choisissez Create (Créer), S3 bucket data (Données du compartiment S3), puis Bulk add columns (Ajout de colonnes en bloc) dans la section Column details (Détails de la colonne).



Tapez des paires nom/valeur dans la zone de texte et choisissez Add.



Améliorations

- Amélioration des performances sur les tables Parquet volumineuses.

Historique du document

Dernière mise à jour de la documentation : 28 juin 2024.

Nous mettons la documentation à jour régulièrement pour prendre en compte vos commentaires. Le tableau ci-dessous décrit les ajouts majeurs apportés à la documentation Amazon Athena. Toutes les mises à jour ne sont pas représentées.

Modification	Description	Date de publication
Politique gérée par `AmazonAthenaFullAccess` mise à jour.	Ajouté `glue:GetCatalogImportStatus` à la `BaseGluePermissions` section de la politique AmazonAthenaFullAccessgérée. L'action ajoutée permet à Athena d'utiliser l' AWS Glue API documentée publiquement pour récupérer le statut d'importation du catalogue.	18 juin 2024
Politique gérée par `AmazonAthenaFullAccess` mise à jour.	Les `datazone:ListAccountEnvironments` autorisations `datazone:ListDomains` `datazone:ListProjects` , et ont été ajoutées à la politique AmazonAthenaFullAccessgérée. Les actions ajoutées permettent aux utilisateurs d'Athena de travailler avec des DataZone domaines, des projets et des environnements Amazon. Pour plus d'informations, consultez Utilisation d'Amazon DataZone dans Athena.	3 janvier 2024
Politique gérée par `AmazonAthenaFullAccess` mise à jour.	Ajout de `glue:StartColumnStatisticsTaskRun` `glue:GetColumnStatisticsTaskRun` , et `glue:GetColumnStatisticsTaskRuns` d'autorisations à la politique AmazonAthenaFullAccessgérée. Les actions ajoutées permettent à Athena d'appeler pour récupérer des statistiques relatives AWS Glue à la fonction d'optimisation basée sur les coûts. Pour plus d'informations, consultez Utilisation de l'optimiseur basé sur les coûts.	3 janvier 2024

Modification	Description	Date de publication
Documentation ajoutée pour les groupes de travail Athena compatibles avec IAM Identity Center.	Vous pouvez créer des groupes de travail Athena SQL utilisant le mode d'authentification IAM Identity Center. Ces groupes de travail prennent en charge l'utilisation de la même identité pour AWS des services tels qu'Amazon Athena et Amazon EMR Studio. Pour plus d'informations, consultez Utilisation des groupes de travail Athena compatibles avec IAM Identity Center.	5 décembre 2023
Documentation ajoutée pour l'interrogation des données S3 Express One Zone	Vous pouvez utiliser Athena pour interroger les données dans la classe de stockage Amazon S3 Express One Zone. Pour plus d'informations, consultez Interrogation des données de S3 Express One Zone.	28 novembre 2023
Documentation ajoutée pour les affichages du Catalogue de données Glue.	Vous pouvez utiliser les affichages du Catalogue de données Glue pour fournir un affichage commun unique de services AWS tels qu'Amazon Athena et Amazon Redshift. Pour plus d'informations, consultez Utilisation des AWS Glue Data Catalog vues.	27 novembre 2023
Documentation ajoutée pour la fonctionnalité d'optimiseur basé sur les coûts.	Vous pouvez utiliser les statistiques de AWS Glue pour optimiser vos requêtes dans Athena SQL. Pour plus d'informations, consultez Utilisation de l'optimiseur basé sur les coûts.	17 novembre 2023
Ajout de documentation pour le pilote Athena JDBC 3.x	Vous pouvez utiliser le pilote Athena JDBC 3.x pour lire les résultats des requêtes directement depuis Amazon S3. Le pilote JDBC 3.x prend en charge presque toutes les méthodes d'authentification prises en charge par le pilote JDBC 2.x. Pour plus d'informations, consultez Pilote Athena JDBC 3.x.	16 novembre 2023

Modification	Description	Date de publication
Ajout de documentation pour une utilisation DataZone dans Athena.	Vous pouvez l'utiliser DataZone pour simplifier votre expérience grâce à des services AWS d'analyse tels qu'Athena et Lake Formation. AWS Glue Pour plus d'informations, consultez Utilisation d'Amazon DataZone dans Athena.	4 octobre 2023
Ajout de la documentation sur les réserves de capacité.	Vous pouvez désormais utiliser les réserves de capacité sur Amazon Athena pour exécuter des requêtes SQL sur une capacité de calcul entièrement gérée. Pour plus d'informations, consultez Gestion de la capacité de traitement des requêtes.	28 avril 2023
Ajout de la documentation sur l'interrogation des vues fédérées.	Vous pouvez désormais créer et interroger des vues sur des sources de données fédérées dans Athena. Pour plus d'informations, consultez Interrogation des vues fédérées.	4 avril 2023
Ajout de la documentation sur la prévention de la limitation dans Amazon S3.	Pour plus d'informations, consultez Prévention de la limitation Amazon S3.	24 mars 2023
Politique gérée par `AmazonAthenaFullAccess` mise à jour.	Ajouté `pricing:GetProducts` à la politique AmazonAthenaFullAccessgérée. L'action ajoutée permet d'accéder à AWS Billing and Cost Management. Pour plus d'informations, consultez GetProductsla référence de AWS Billing and Cost Management l'API.	25 janvier 2023
Documentation étoffée sur la prise en charge de la compression Athena.	Rubriques individuelles ajoutées pour Compression de la table Hive, compression de la table Iceberg et Niveaux de compression ZSTD. Pour plus d'informations, consultez Prise en charge de la compression Athena.	20 janvier 2023

Modification	Description	Date de publication
Ajout de documentation pour Amazon Athena pour Apache Spark.	Vous pouvez désormais créer et exécuter de manière interactive des applications Apache Spark et des blocs-notes compatibles avec Jupyter sur Amazon Athena. Pour plus d'informations, consultez Utilisation d'Apache Spark dans Amazon Athena.	30 novembre 2022
Ajout de documentation pour le connecteur Db2 IBM d'Athena.	Vous pouvez utiliser le connecteur Amazon Athena pour Db2 IBM pour interroger Db2 depuis Athena. Pour plus d'informations, consultez Connecteur Amazon Athena pour Db2 IBM.	18 novembre 2022
Ajout de documentation pour la réutilisation des résultats des requêtes.	Lorsque vous réexécutez une requête dans Athena, vous pouvez désormais choisir de réutiliser le dernier résultat stocké de la requête. Cela peut augmenter les performances et réduire les coûts en termes de nombre d'octets analysés. Pour plus d'informations, consultez Réutilisation des résultats des requêtes.	8 novembre 2022
Documentation mise à jour pour CloudTrail les journaux.	Le `CREATE TABLE` DDL pour interroger les CloudTrail journaux a été mis à jour pour utiliser le JSON SerDe au lieu du. CloudTrail SerDe Pour plus d'informations, consultez Journaux d'interrogation AWS CloudTrail.	3 novembre 2022
Ajout de la documentation relative à la version 3 du moteur Athena.	Pour plus d'informations sur la version 3 du moteur Athena, consultez la rubrique Version 3 du moteur Athena.	13 octobre 2022
Ajout d'un tutoriel sur la configuration de SSO pour ODBC à l'aide du plugin Okta.	Configurer le pilote ODBC Amazon Athena et le plugin Okta pour la fonctionnalité d'authentification unique (SSO) en utilisant le fournisseur d'identité Okta. Pour plus d'informations, consultez Configuration de SSO pour ODBC en utilisant le plugin Okta et le fournisseur d'identité Okta.	23 août 2022

Modification	Description	Date de publication
Ajout d'une documentation pour la visualisation des plans de requêtes et des statistiques dans la console Athena.	Vous pouvez utiliser l'éditeur de requêtes Athena pour voir des représentations graphiques de la manière dont vos requêtes seront exécutées, ainsi que des graphiques, des détails et des statistiques sur la manière dont les requêtes terminées ont été exécutées. Pour plus d'informations, consultez Affichage des plans d'exécution pour les requêtes SQL et Affichage des statistiques et des détails d'exécution pour les requêtes terminées.	21 juillet 2022
Ajout de la documentation pour interroger les vues Apache Hive dans les métastores Hive externes.	Vous pouvez utiliser Athena pour interroger les vues Apache créées dans des métastores Hive externes. Certaines fonctions Hive ne sont pas prises en charge ou nécessitent un traitement spécial. Pour plus d'informations, consultez Utilisation des vues Hive.	22 avril 2022
Ajout de documentation relative aux requêtes enregistrées.	Vous pouvez utiliser la fonction de requêtes enregistrées dans Athena pour enregistrer, rappeler, modifier et renommer vos requêtes. Pour plus d'informations, consultez Utilisation de requêtes enregistrées ce guide et UpdateNamedQueryle manuel Amazon Athena API Reference.	28 février 2022
Documentation d'aperçu ajoutée pour la prise en charge d'Apache Iceberg.	Athena prend en charge les requêtes de lecture, de voyage dans le temps et d'écriture pour les tables Apache Iceberg qui utilisent le format Apache Parquet pour les données et le AWS Glue catalogue pour leur métastore. Pour plus d'informations, consultez Utilisation des tables Apache Iceberg.	26 novembre 2021

Modification	Description	Date de publication
Documentation ajoutée pour les requêtes fédérées entre comptes.	Vous pouvez utiliser la fonction de requête fédérée entre comptes pour interroger des sources de données dans un autre compte. Pour plus d'informations sur la configuration des autorisations permettant d'activer cette fonction, consultez Activation des requêtes fédérées entre comptes.	12 novembre 2021
Ajout de documentation pour l'instruction `UNLOAD` d'Athena.	Utilisation de l'instruction `UNLOAD` pour écrire la requête des résultats à partir d'une instruction `SELECT` aux formats Apache Parquet, ORC, Apache Avro et JSON. Pour plus d'informations, consultez UNLOAD.	5 août 2021
Ajout de documentation pour la fonction d'instruction `EXPLAIN` d'Athena.	Pour plus d'informations, consultez Utilisation de EXPLAIN et EXPLAIN ANALYZE sur Athena et Explication des résultats de l'instruction EXPLAIN d'Athena.	5 avril 2021
Ajout de pages sur le dépannage et le réglage des performances dans Athena.	Pour plus d'informations, consultez Résolution des problèmes dans Athena et Réglage de performances dans Athena.	30 décembre 2020
Ajout de la documentation relative à la gestion des versions du moteur Athena et à la version 2 du moteur Athena.	Pour plus d'informations, consultez Gestion des versions du moteur Athena.	11 novembre 2020

Modification	Description	Date de publication
Mise à jour de la documentation sur les requêtes fédérées pour la version de disponibilité générale.	Pour plus d'informations, consultez Utilisation de la requête fédérée d'Amazon Athena et Utilisation d'Athéna avec CalledVia des touches contextuelles.	11 novembre 2020
Ajout de la documentation pour l'utilisation du pilote JDBC avec Lake Formation pour l'accès fédéré à Athena.	Pour plus d'informations, consultez Utilisation de Lake Formation et des pilotes JDBC et ODBC d'Athena pour l'accès fédéré à Athena et Tutoriel : configuration de l'accès fédéré des utilisateurs d'Okta à Athena en utilisant Lake Formation et JDBC.	25 septembre 2020
Ajout de documentation pour le connecteur de OpenSearch données Amazon Athena.	Pour plus d'informations, consultez Connecteur Amazon Athena OpenSearch .	21 juillet 2020
Ajout de la documentation pour interroger les jeux de données Hudi.	Pour plus d'informations, consultez Utilisation d'Athena pour interroger des jeux de données Apache Hudi.	9 juillet 2020

Modification	Description	Date de publication
Ajout de la documentation sur l'interrogation des journaux du serveur Web Apache et des journaux du serveur Web IIS stockés dans Simple Storage Service (Amazon S3).	Pour plus d'informations, consultez Interrogation des journaux Apache stockés dans Simple Storage Service (Amazon S3) et Interrogation de journaux Internet Information Server (IIS) stockés dans Simple Storage Service (Amazon S3).	8 juillet 2020
Ajout de la documentation relative à la version générale du connecteur de données Athena pour le métastore Hive externe.	Pour plus d'informations, consultez Utilisation du connecteur de données Athena pour un métastore Hive externe.	1er juin 2020
Ajout de la documentation pour le balisage des ressources du catalogue de données.	Pour plus d'informations, consultez Étiquetage des ressources Athena.	1er juin 2020
Ajout de documentation sur la projection de partition.	Pour plus d'informations, consultez Projection de partition avec Amazon Athena.	21 mai 2020

Modification	Description	Date de publication
Mise à jour des exemples de code Java pour Athena.	Pour plus d'informations, consultez Exemples de code.	11 mai 2020
Ajout d'une rubrique sur l'interrogation des GuardDuty résultats d'Amazon.	Pour plus d'informations, consultez Interroger les résultats d'Amazon GuardDuty .	19 mars 2020
Ajout d'une rubrique sur l'utilisation CloudWatch des événements pour surveiller les transitions d'état des requêtes Athena.	Pour plus d'informations, consultez Surveillance des requêtes Athena à l'aide des événements Amazon EventBridge.	11 mars 2020
Ajout d'une rubrique sur l'interrogation des journaux de AWS Global Accelerator flux avec Athena.	Pour plus d'informations, consultez Interrogation des journaux AWS Global Accelerator de flux.	6 février 2020

Modification	Description	Date de publication
• Ajout de la documentation sur l'utilisation de CTAS avec INSERT INTO pour ajouter des données à partir d'une source non partitionnée vers une destination partitionnée. • Ajout de liens de téléchargement pour la version de prévisualisation 1.1.0 du pilote ODBC pour Athena. • Description de l'expression régulière `SHOW DATABASES LIKE` corrigée. • Syntaxe `partitioned_by` corrigée dans la rubrique CTA. • Autres corrections mineures.	Les mises à jour de la documentation incluent, sans s'y limiter, les rubriques suivantes : • Utilisation de CTAS et INSERT INTO pour ETL et l'analyse des données • Connexion à Amazon Athena avec le pilote ODBC (Les fonctions de prévisualisation 1.1.0 sont maintenant incluses dans le pilote ODBC 1.1.2.) • SHOW DATABASES • CREATE TABLE AS	4 février 2020

Modification	Description	Date de publication
Ajout de la documentation sur l'utilisation de CTAS avec INSERT INTO pour ajouter des données d'une source partitionnée vers une destination partitionnée.	Pour plus d'informations, consultez Utilisation de CTAS et de INSERT INTO pour contourner la limite de 100 partitions.	22 janvier 2020
Informations de localisation des résultats de requête mises à jour.	Athena ne crée plus un emplacement de résultats de requête « par défaut ». Pour plus d'informations, consultez Spécification d'un emplacement de résultats de requête.	20 janvier 2020
Ajout d'un sujet sur l'interrogation du. AWS Glue Data Catalog Mise à jour des informations sur les quotas de service (anciennement « limites de service ») dans Athena.	Pour plus d'informations, consultez les rubriques suivantes : • Interrogation du AWS Glue Data Catalog • Service Quotas	17 janvier 2020

Modification	Description	Date de publication
Rubrique corrigée sur SerDe OpenCSV pour noter que `TIMESTAMP` le type doit être spécifié au format numérique UNIX.	Pour plus d'informations, consultez SerDe OpenCSV pour le traitement des fichiers CSV.	15 janvier 2020
Mise à jour de la rubrique de sécurité sur le chiffrement pour noter qu'Athena ne prend pas en charge les clés asymétriques.	Athena ne prend en charge que les clés symétriques pour la lecture et l'écriture de données. Pour plus d'informations, consultez Options de chiffrement Simple Storage Service (Amazon S3) prises en charge.	8 janvier 2020
Ajout d'informations sur l'accès entre comptes aux compartiments Amazon S3 chiffrés à l'aide d'une clé personnalisée AWS KMS .	Pour plus d'informations, consultez Accès entre comptes à un compartiment chiffré à l'aide d'une clé personnalisée AWS KMS.	13 décembre 2019

Modification	Description	Date de publication
Ajout de la documentation pour les requêtes fédérées, les métastores Hive externes, le machine learning et les fonctions définies par l'utilisateur. Ajout de nouvelles métriques CloudWatch.	Pour plus d'informations, consultez les rubriques suivantes : • Utilisation de la requête fédérée d'Amazon Athena • Connecteurs de source de données disponibles • Utilisation du connecteur de données Athena pour un métastore Hive externe • Utilisation du Machine Learning (ML) avec Amazon Athena • Interrogation avec des fonctions définies par l'utilisateur • Liste des CloudWatch métriques et des dimensions d'Athena	26 novembre 2019
Ajout d'une section pour la nouvelle commande INSERT INTO et mise à jour des informations d'emplacement des résultats de requête pour la prise en charge des fichiers manifestes de données.	Pour plus d'informations, consultez INSERT INTO et Utilisation des résultats des requêtes, des requêtes récentes et des fichiers de sortie.	18 septembre 2019

Modification	Description	Date de publication
Ajout d'une section pour le support des points de terminaison VPC d'interface ()PrivateLink. Mise à jour des pilotes JDBC. Mise à jour des informations sur les journaux de flux VPC enrichi.	Pour plus d'informations, consultez Connexion à Amazon Athena à l'aide d'un point de terminaison de VPC d'interface, Interrogation des journaux de flux Amazon VPC et Connexion à Amazon Athena avec JDBC.	11 septembre 2019
Ajout d'une section sur l'intégration avec AWS Lake Formation.	Pour plus d'informations, consultez Utilisation d'Athena pour interroger des données enregistrées dans AWS Lake Formation.	26 juin 2019
Mise à jour de la section Sécurité pour plus de cohérence avec les autres services AWS .	Pour plus d'informations, consultez Sécurité Amazon Athena.	26 juin 2019
Ajout d'une section sur l'interrogation des AWS WAF journaux.	Pour plus d'informations, consultez Journaux d'interrogation AWS WAF.	31 mai 2019

Modification	Description	Date de publication
Publication de la nouvelle version du pilote ODBC avec prise en charge des groupes de travail Athena.	Pour télécharger le pilote ODBC version 1.0.5 et sa documentation, consultez Connexion à Amazon Athena avec le pilote ODBC. Aucune modification apportée à la chaîne de connexion du pilote ODBC lorsque vous utilisez des identifications sur des groupes de travail. Pour utiliser des identifications, mettez à niveau vers la dernière version du pilote ODBC qui est la version actuelle. Ce pilote vous permet d'utiliser des actions de groupe de travail d'API Athena pour créer et gérer des groupes de travail, et des actions d'étiquetage d'API Athena pour ajouter, répertorier ou supprimer des étiquettes sur les groupes de travail. Avant de commencer, veillez à disposer des autorisations au niveau des ressources dans IAM pour exécuter des actions sur les groupes de travail et des étiquettes.	5 mars 2019
Ajout de la prise en charge des étiquettes pour les groupes de travail dans Amazon Athena.	une identification est constituée d'une clé et d'une valeur que vous définissez. Lorsque vous identifiez un groupe de travail, vous lui attribuez des métadonnées personnalisées. Par exemple, créez un groupe de travail pour chaque centre de coûts. Ensuite, en ajoutant des étiquettes à ces groupes de travail, vous pouvez suivre vos dépenses Athena pour chaque centre de coûts. Pour plus d'informations, consultez Utilisation d'identifications pour la facturation dans le Guide de l'utilisateur AWS Billing and Cost Management .	22 février 2019

Modification	Description	Date de publication
Amélioration du JSON OpenX SerDe utilisé dans Athena.	Ces améliorations incluent, sans toutefois s'y limiter : • Prise en charge de la propriété `ConvertDots InJsonKeysToUnderscores` . Lorsqu'il est défini sur`TRUE`, il permet de SerDe remplacer les points dans les noms clés par des traits de soulignement. Par exemple, si le jeu de données JSON contient une clé portant le nom `"a.b"`, vous pouvez utiliser cette propriété pour définir le nom de la colonne comme étant `"a_b"` dans Athena. L'argument par défaut est `FALSE`. Par défaut, Athena n'autorise pas les points dans les noms de colonnes. • Prise en charge de la propriété `case.insensitive` . Par défaut, Athena exige que toutes les clés de votre jeu de données JSON soient en minuscules. `WITH SERDE PROPERTIES ("case.insensitive"= FALSE;)` vous permet d'utiliser des noms de clé sensibles à la casse dans vos données. L'argument par défaut est `TRUE`. Lorsqu'il est défini sur`TRUE`, il SerDe convertit toutes les colonnes majuscules en minuscules. Pour plus d'informations, consultez OpenX JSON SerDe.	18 février 2019

Modification	Description	Date de publication
Ajout de la prise en charge des groupes de travail.	Utilisation de groupes de travail pour séparer les utilisateurs, les équipes, les applications ou les charges de travail, et pour définir des limites au volume de données pouvant être traité par chaque requête ou groupe de travail entier. Vous pouvez utiliser des autorisations au niveau des ressources IAM pour contrôler l'accès à un groupe de travail spécifique, car les groupes de travail agissent en tant que ressources IAM. Vous pouvez également consulter les métriques relatives aux requêtes dans Amazon CloudWatch, contrôler les coûts des requêtes en limitant la quantité de données numérisées, créer des seuils et déclencher des actions, telles que des alarmes Amazon SNS, lorsque ces seuils sont dépassés. Pour plus d'informations, consultez Utilisation de groupes de travail pour exécuter des requêtes et Contrôle des coûts et surveillance des requêtes à l'aide de CloudWatch métriques et d'événements.	18 février 2019
Ajout de la prise en charge de l'analyse des journaux du Network Load Balancer.	Ajout d'exemples de requêtes Athena pour l'analyse des journaux du Network Load Balancer. Ces journaux reçoivent des informations détaillées sur les demandes TLS (Transport Layer Security, Sécurité de la couche de transport) envoyées au Network Load Balancer. Vous pouvez utiliser ces journaux d'accès pour analyser les modèles de trafic et résoudre des problèmes. Pour plus d'informations, veuillez consulter the section called "Network Load Balancer".	24 janvier 2019

Modification	Description	Date de publication
Lancement des nouvelles versions des pilotes JDBC et ODBC avec prise en charge de l'accès fédéré à l'API Athena avec AD FS et SAML 2.0 (Security Assertion Markup Language 2.0).	Avec cette version des pilotes, l'accès fédéré à Athena est pris en charge pour Active Directory Federation Service (AD FS 3.0). L'accès est établi via les versions des pilotes JDBC ou ODBC prenant en charge SAML 2.0. Pour en savoir plus sur la configuration de l'accès fédéré à l'API Athena, voir the section called “Activation de l'accès fédéré à l'API Athena”.	10 novembre 2018
Ajout de la prise en charge du contrôle précis des accès aux bases de données et tables dans Athena. En outre, ajout dans Athena de politiques qui vous permettent de chiffrer les métadonnées des bases de données et des tables dans le catalogue de données.	Ajout de la prise en charge de la création de politiques basées sur l'identité (IAM) qui fournissent un contrôle d'accès précis aux ressources du AWS Glue Data Catalog, telles que les bases de données et les tables utilisées dans Athena. De plus, vous pouvez chiffrer les métadonnées de bases de données et de tables dans le catalogue de données, en ajoutant des politiques spécifiques à Athena. Pour plus de détails, consultez Accès détaillé aux bases de données et aux tables du AWS Glue Data Catalog.	15 octobre 2018

Modification	Description	Date de publication
Ajout de la prise en charge des instructions `CREATE TABLE AS SELECT`. Autres améliorations apportées à la documentation.	Ajout de la prise en charge des instructions `CREATE TABLE AS SELECT`. Consultez Création d'une table à partir des résultats des requêtes (CTAS), Considérations et limitations relatives aux requêtes CTAS et Exemples de requêtes CTAS.	10 octobre 2018
Publication de la version du pilote ODBC 1.0.3 avec prise en charge des résultats de streaming au lieu de les extraire des pages. Autres améliorations apportées à la documentation.	La version du pilote ODBC 1.0.3 prend en charge les résultats de streaming et inclut également des améliorations, des correctifs de bogues et d'une mise à jour de la documentation pour « Utilisation de SSL avec un serveur proxy ». Pour plus d'informations sur le téléchargement du pilote ODBC version 1.0.3 et de sa documentation, consultez Connexion à Amazon Athena avec le pilote ODBC.	6 septembre 2018

Modification	Description	Date de publication
Publication de la version du pilote JDBC 2.0.5 avec prise en charge par défaut des résultats de streaming au lieu de les extraire des pages. Autres améliorations apportées à la documentation.	Publication de la version du pilote JDBC 2.0.5 avec prise en charge par défaut des résultats de streaming au lieu de les extraire des pages. Pour plus d'informations, veuillez consulter Connexion à Amazon Athena avec JDBC.	16 août 2018
Mise à jour de la documentation pour l'interrogation des flux de journaux de cloud privé virtuel Amazon Virtual Private Cloud, qui peuvent être stockés directement dans Simple Storage Service (Amazon S3) dans un format GZIP. Mise à jour des exemples pour l'interrogation des journaux ALB.	Mise à jour de la documentation pour l'interrogation des flux de journaux de cloud privé virtuel Amazon Virtual Private Cloud, qui peuvent être stockés directement dans Simple Storage Service (Amazon S3) dans un format GZIP. Pour plus d'informations, veuillez consulter Interrogation des journaux de flux Amazon VPC. Mise à jour des exemples pour l'interrogation des journaux ALB. Pour plus d'informations, veuillez consulter Interrogation des journaux de l'Application Load Balancer.	7 août 2018

Modification	Description	Date de publication
Ajout de la prise en charge des vues. Ajout de recommandations pour les manipulations de schéma pour différents formats de stockage de données.	Ajout de la prise en charge des vues. Pour plus d'informations, veuillez consulter Utilisation des vues. Mise à jour de ce guide avec des conseils sur le traitement des mises à jour de schéma pour divers formats de stockage de données. Pour plus d'informations, veuillez consulter Traitement des mises à jour de schéma.	5 juin 2018
Augmentation des limites de simultanéité par défaut des requêtes de cinq à vingt.	Vous pouvez soumettre et exécuter jusqu'à vingt `DDL` requêtes et vingt `SELECT` requêtes en même temps. Pour plus d'informations, veuillez consulter Service Quotas.	17 mai 2018
Ajout d'onglets de requête, et d'une possibilité de configurer les informations automatiques dans l'Editeur de requête.	Ajout d'onglets de requête, et d'une possibilité de configurer les informations automatiques dans l'Editeur de requête. Pour plus d'informations, veuillez consulter Mise en route.	8 mai 2018
Publication du pilote JDBC version 2.0.2.	Publication de la nouvelle version du pilote JDBC (version 2.0.2). Pour plus d'informations, veuillez consulter Connexion à Amazon Athena avec JDBC.	19 avril 2018

Modification	Description	Date de publication
Ajout d'une fonction de remplissage automatique pour la saisie des requêtes dans la console Athena.	Ajout d'une fonction de remplissage automatique pour la saisie des requêtes dans la console Athena.	6 avril 2018
Ajout de la possibilité de créer des tables Athena pour les fichiers CloudTrail journaux directement depuis la CloudTrail console.	Ajout de la possibilité de créer automatiquement des tables Athena pour les fichiers CloudTrail journaux directement depuis la CloudTrail console. Pour plus d'informations, veuillez consulter Utilisation de la CloudTrail console pour créer une table Athena pour les journaux CloudTrail .	15 mars 2018
Ajout de la prise en charge du déchargement sécurisé des données intermédiaires sur disque pour les requêtes avec `GROUP BY`.	Ajout de la possibilité de décharger en toute sécurité des données intermédiaires sur le disque pour les requêtes nécessitant beaucoup de mémoire qui utilisent la clause `GROUP BY`. Cela permet d'améliorer la fiabilité de ces requêtes et empêche les erreurs liées à l'épuisement des ressources de requête. Pour plus d'informations, consultez les notes de mise à jour de 2 février 2018.	2 février 2018
Ajout de la prise en charge de Presto version 0.172.	Mise à niveau du moteur sous-jacent dans Amazon Athena vers une version basée sur Presto version 0.172. Pour plus d'informations, consultez les notes de mise à jour de 19 janvier 2018.	19 janvier 2018
Ajout de la prise en charge du pilote ODBC.	Ajout de la prise en charge de la connexion d'Athena au pilote ODBC. Pour obtenir des informations, consultez Connexion à Amazon Athena avec ODBC.	13 novembre 2017

Modification	Description	Date de publication
Ajout de la prise en charge des régions Asie-Pacifique (Séoul), Asie-Pacifique (Mumbai) et Europe (Londres). Ajout de la prise en charge des requêtes de données géospatiales.	Ajout de la prise en charge de l'interrogation des données géospatiales et des régions Asie-Pacifique (Séoul), Asie-Pacifique (Mumbai) et Europe (Londres). Pour plus d'informations, consultez la rubrique Interrogation des données géospatiales et Régions AWS et points de terminaison.	1 novembre 2017
Ajout de la prise en charge de l'Europe (Francfort).	Ajout de la prise en charge de l'Europe (Francfort). Pour obtenir la liste des régions prises en charge, consultez Régions AWS et points de terminaison.	19 octobre 2017
Ajout du support pour les requêtes Athena nommées avec. AWS CloudFormation	Ajout de la prise en charge de la création de requêtes Athena nommées avec. AWS CloudFormation Pour plus d'informations, consultez AWS::Athena::NamedQueryle guide de AWS CloudFormation l'utilisateur.	3 octobre 2017
Ajout de la prise en charge de l'Asie-Pacifique (Sydney).	Ajout de la prise en charge de l'Asie-Pacifique (Sydney). Pour obtenir la liste des régions prises en charge, consultez Régions AWS et points de terminaison.	25 septembre 2017

Modification	Description	Date de publication
Ajout d'une section à ce guide pour interroger les Service AWS journaux et différents types de données, notamment les cartes, les tableaux, les données imbriquées et les données contenant du JSON.	Ajout d'exemples pour Journaux d'interrogation Service AWS et pour l'interrogation des différents types de données dans Athena. Pour plus d'informations, veuillez consulter Exécution de requêtes SQL à l'aide d'Amazon Athena.	5 septembre 2017
Ajout du support pour AWS Glue Data Catalog.	Ajout d'une intégration avec le AWS Glue Data Catalog et d'un assistant de migration pour la mise à jour du catalogue de données géré Athena vers le. AWS Glue Data Catalog Pour plus d'informations, consultez la rubrique Intégration dans AWS Glue et AWS Glue.	14 août 2017
Ajout du support pour Grok. SerDe	Ajout de la prise en charge de Grok SerDe, qui facilite la correspondance des modèles pour les enregistrements dans des fichiers texte non structurés tels que les journaux. Pour plus d'informations, consultez Grok. SerDe Ajout de raccourcis clavier pour parcourir l'historique des requêtes à l'aide de la console.	4 août 2017
Ajout de la prise en charge de la région Asie-Pacifique (Tokyo).	Ajout de la prise en charge des régions Asie-Pacifique (Tokyo) et Asie-Pacifique (Singapour). Pour obtenir la liste des régions prises en charge, consultez Régions AWS et points de terminaison.	22 juin 2017

Modification	Description	Date de publication
Ajout de la prise en charge de l'Europe (Irlande).	Ajout de la prise en charge de l'Europe (Irlande). Pour plus d'informations, consultez Régions AWS et points de terminaison.	8 juin 2017
Ajout d'une API et AWS CLI d'un support Amazon Athena.	Ajout d'une API Amazon Athena et du AWS CLI support pour Athena. Mise à jour du pilote JDBC vers la version 1.1.0.	19 mai 2017
Ajout de la prise en charge du chiffrement des données Simple Storage Service (Amazon S3).	Ajout de la prise en charge du chiffrement des données Simple Storage Service (Amazon S3) et publication de la mise à jour d'un pilote JDBC (version 1.0.1) avec des améliorations de la prise en charge du chiffrement et des corrections de bogues. Pour plus d'informations, consultez Chiffrement au repos.	4 avril 2017
A ajouté le AWS CloudTrail SerDe.	Ajout de l' AWS CloudTrail SerDeamélioration des performances, résolution des problèmes de partition. • Le AWS CloudTrail SerDe a été remplacé par le Hive JSON SerDe pour lire CloudTrail les journaux. Pour plus d'informations sur l'interrogation CloudTrail des journaux, consultezJournaux d'interrogation AWS CloudTrail. • Amélioration des performances lors de l'analyse d'un grand nombre de partitions. • Amélioration des performances sur l'opération `MSCK Repair Table`. • Ajout de la possibilité d'exécuter des requêtes sur les données Amazon S3 stockées dans des régions autres que votre région principale. Les taux standard de transfert de données entre régions pour Simple Storage Service (Amazon S3) s'appliquent en plus des frais Athena standard.	24 mars 2017

Modification	Description	Date de publication
Ajout de la prise en charge de la région USA Est (Ohio).	Ajout de la prise en charge de SerDe Avro et SerDe OpenCSV pour le traitement des fichiers CSV, de la région USA Est (Ohio) et de la modification en masse des colonnes dans l'assistant de la console. Amélioration des performances sur les tables Parquet volumineuses.	20 février 2017
	Version initiale du Guide de l'utilisateur Amazon Athena.	Novembre 2016

AWS Glossaire

Pour la AWS terminologie la plus récente, consultez le AWS glossaire dans la Glossaire AWS référence.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.